Learning CRFs with Hierarchical Features: An Application to Go

Scott Sanner – University of Toronto

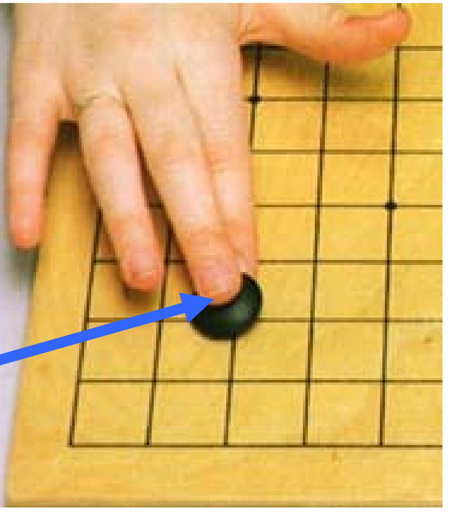
Thore Graepel
Ralf Herbrich
Tom Minka

Microsoft Research

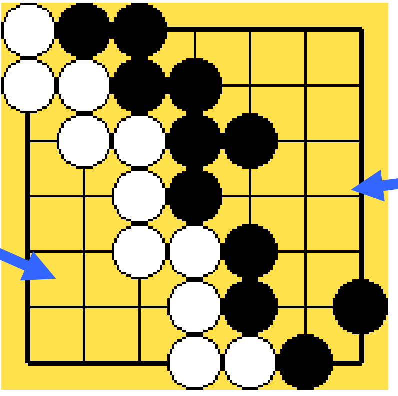
(with thanks to David Stern and Mykel Kochenderfer)

The Game of Go

- Started about 4000 years ago in ancient China
- About 60 million players worldwide
- 2 Players: Black and White
- Board: 19×19 grid
- Rules:
  - Turn: One stone placed on vertex.
  - Capture.
- Aim: Gather territory by surrounding it



White territory

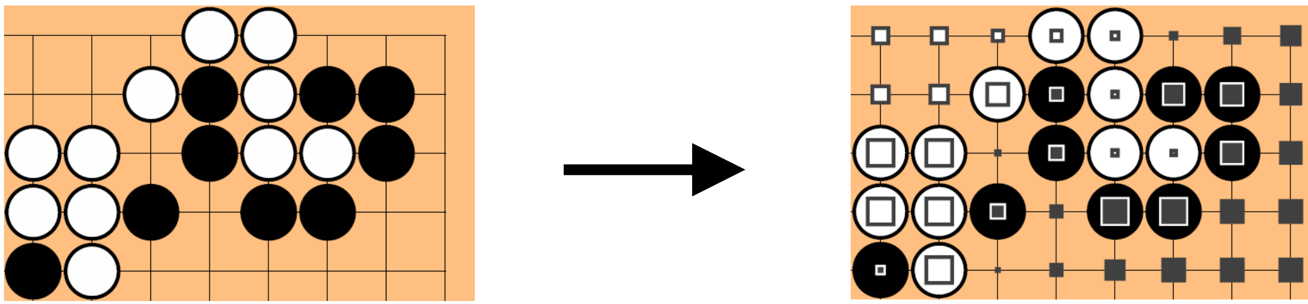


Black territory

Territory Prediction

- Goal: predict territory distribution given board...

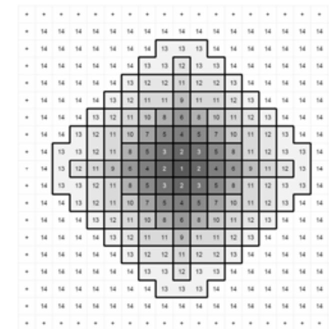
$\vec{c}$ $\qquad\qquad\qquad\qquad$ $P(\vec{s}\,|\,\vec{c})$



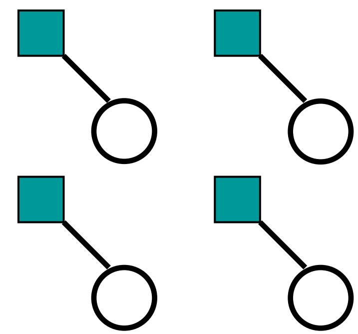
- How to predict territory?
  - Could use search:
    - Full α-β impractical (avg. 180 moves/turn, >100 moves/game)
    - *Monte Carlo* averaging a reasonable estimator, but costly
  - We learn to directly predict territory:
    - Learn $P(\vec{s}\,|\,\vec{c})$ from expert data

Talk Outline

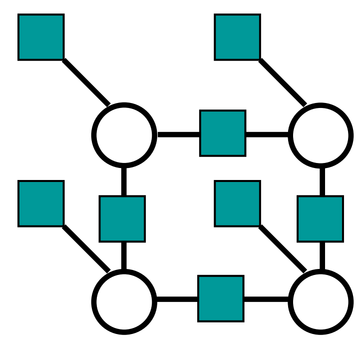
- Hierarchical pattern features



- Independent classifiers
  - What is best way to combine features?



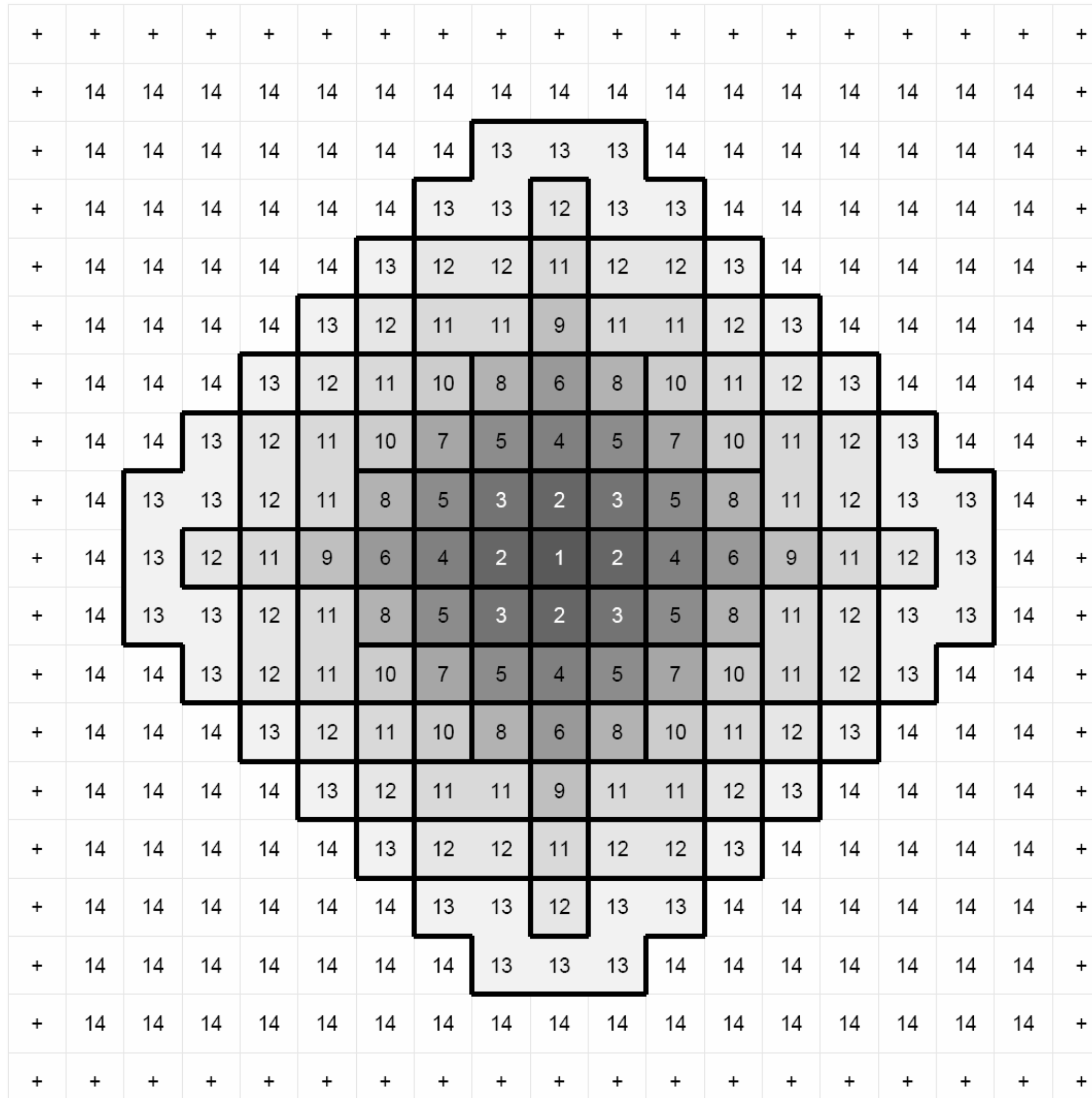
- CRF models
  - Coupling factor model (w/ patterns)
  - Exact inference intractable
  - What is best training / inference method to circumvent intractability?



- Evaluation and Conclusions

Hierarchical Patterns

- Centered on a single position
- Exact config. of stones
- Fixed match region (template)
- 8 nested templates
- 3.8 million patterns mined

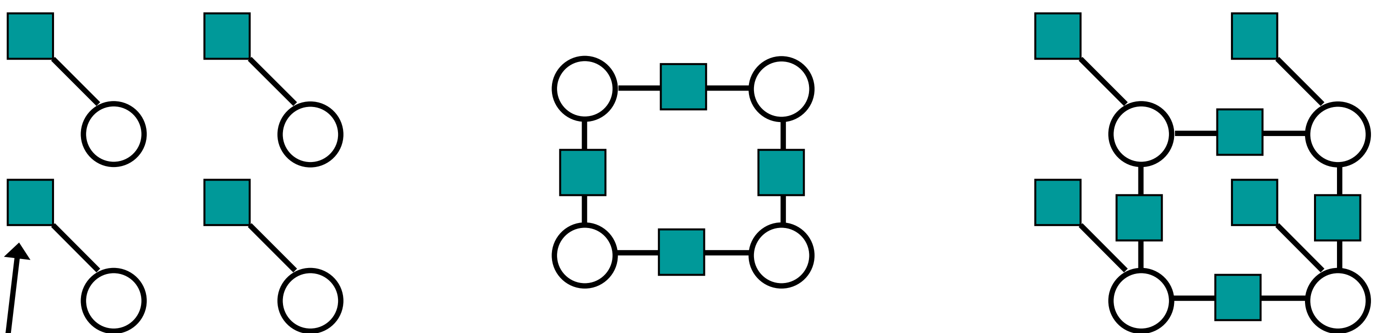


Models

(a) Independent pattern-based classifiers

(b) (Coupling) CRF

(c) Pattern CRF



Vertex Variables: s_i, c_i

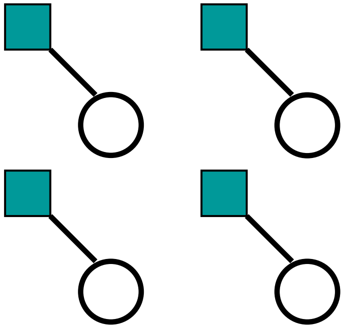
Unary pattern-based factors:

$$\psi_i(s_i = +1, \vec{c}) = \exp\left(\sum_{\tilde{\pi} \in \tilde{\Pi}} \lambda_{\tilde{\pi}} \cdot \mathbb{I}_{\tilde{\pi}}(\vec{c}, i)\right)$$

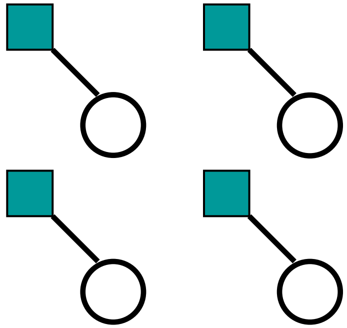
Coupling factors:

$$\psi_{(i,j)}(s_i, s_j, c_i, c_j) = \exp\left(\sum_{k=1}^{36} \lambda_k \cdot \mathbb{I}_k(s_i, s_j, c_i, c_j, k)\right)$$

Independent Pattern-based Classifiers



Inference and Training



- Up to 8 patterns may match at any vertex
- Which pattern to use?
  - Smallest pattern
  - Largest pattern

$$\psi_i(s_i = +1, \vec{c}) = P(s_i | \tilde{\pi}_{\text{min}}(\vec{c}, i))$$
$$\psi_i(s_i = +1, \vec{c}) = P(s_i | \tilde{\pi}_{\text{max}}(\vec{c}, i))$$

- Or, combine all patterns:
  - Logistic regression
  - Bayesian model averaging…

$$\psi_i(s_i = +1, \vec{c})$$
$$= \exp\left(\sum_{\tilde{\pi} \in \tilde{\Pi}} \lambda_{\tilde{\pi}} \cdot \mathbb{I}_{\tilde{\pi}}(\vec{c}, i)\right)$$

Bayesian Model Averaging

- Bayesian approach to combining models:

$$P(s_j|\vec{c}, D) = \sum_{\tau \in \Upsilon} P(s_j|\tau, \vec{c}, D) P(\tau|\vec{c}, D)$$

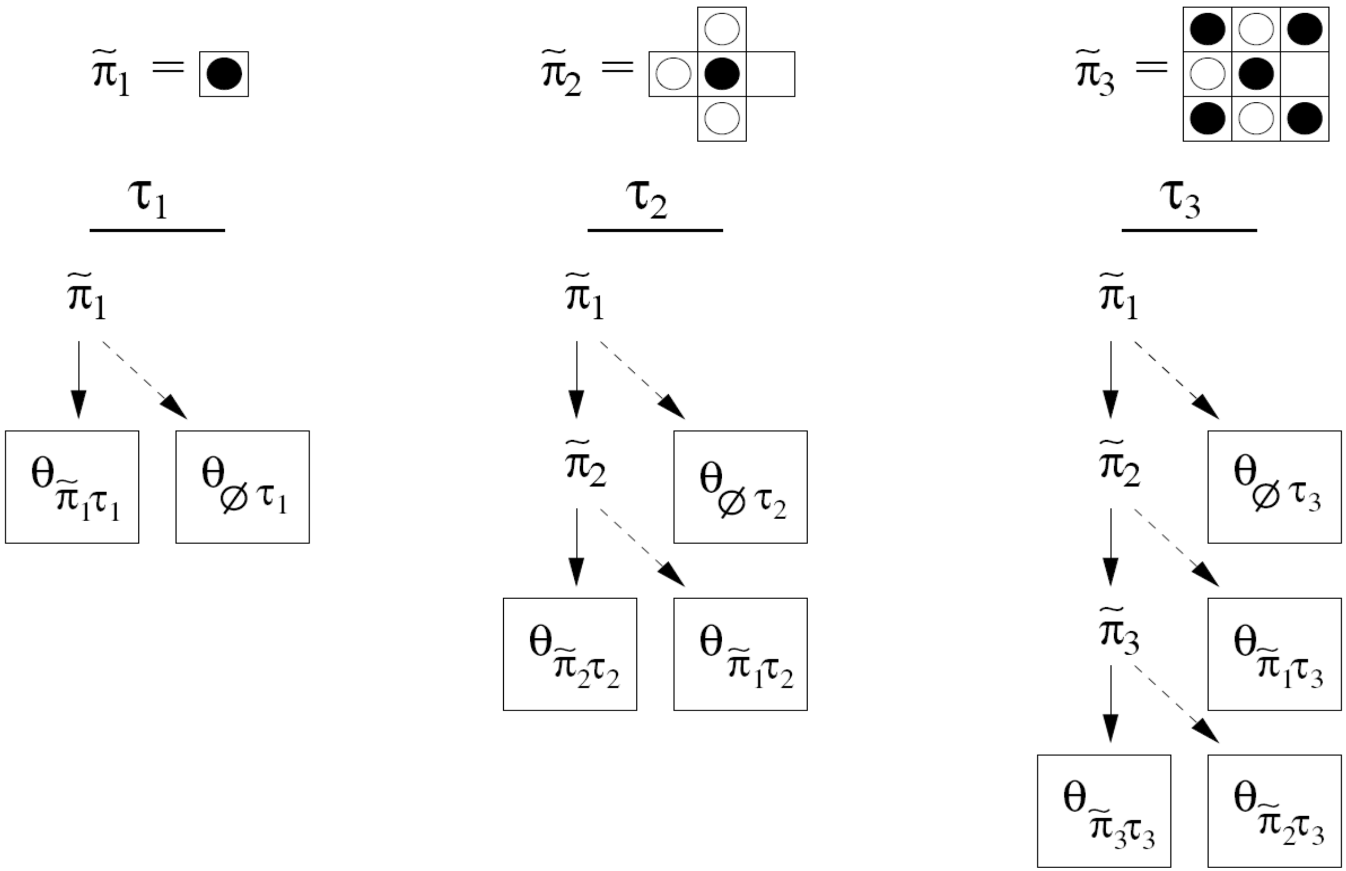
- Now examine the model "weight":

$$P(\tau|\vec{c}, D) = \frac{P(D|\tau, \vec{c}) P(\tau|\vec{c})}{\sum_{\tau \in \Upsilon} P(D|\tau, \vec{c}) P(\tau|\vec{c})}$$

- Model τ must apply to all data!

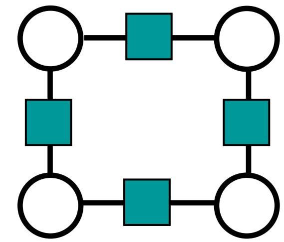
Hierarchical Tree Models

- Arrange patterns into decision trees τ:

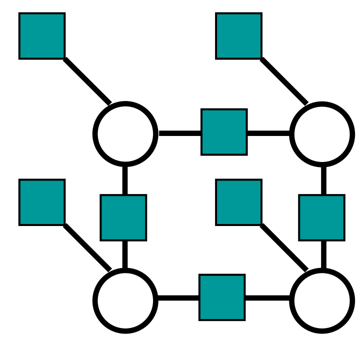


- Model τ provides predictions on all data

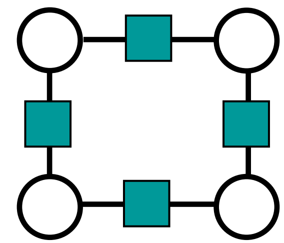
Coupling CRF

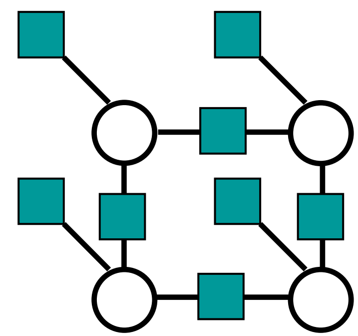


& Pattern CRF



Inference and Training





- Inference
  - Intractable for 19x19 grids
  - Loopy BP is biased
    - but an option
  - Sampling is unbiased
    - but much slower
- Training
  - Max likelihood requires inference!

$$\frac{\partial l}{\partial \lambda_j} = \sum_{d\in D}\left(\mathbb{I}_j(\vec{s}^{(d)},\vec{c}^{(d)}) - \sum_{\vec{s}} \mathbb{I}_j(\vec{s},\vec{c}^{(d)})P(\vec{s}|\vec{c}^{(d)})\right)$$

  - Other approximate methods…

Pseudolikelihood

- Standard log likelihood:

$$l(\vec{\lambda}) = \sum_{d \in D} \log P(\vec{s}^{(d)} | \, \vec{c}^{(d)})$$

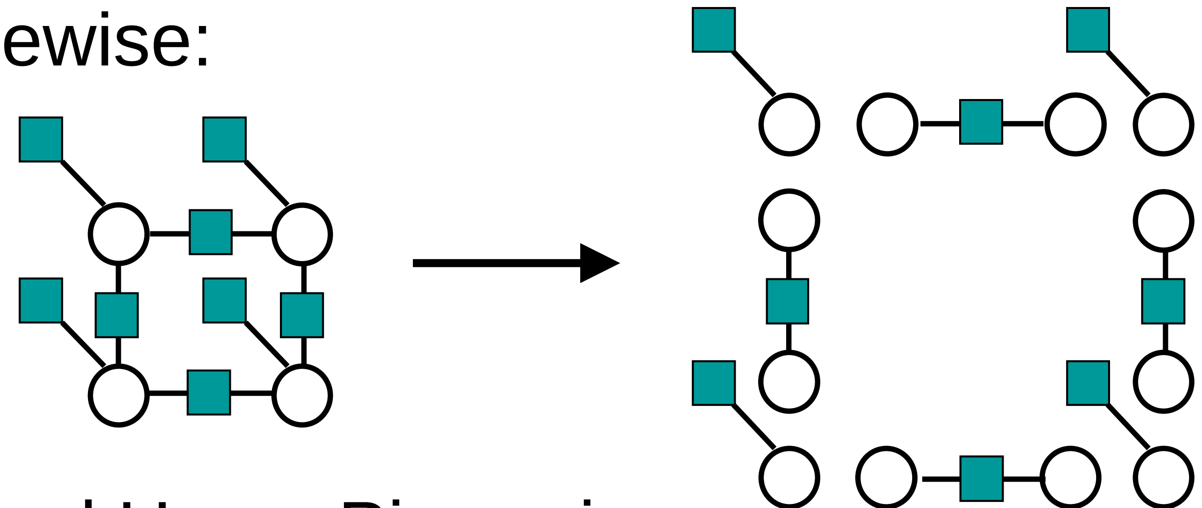
- Pseudo log-likelihood (clamp Markov blanket):

$$pl(\vec{\lambda}) = \sum_{d \in D} \sum_{f \in \mathcal{F}} \log P(\vec{s}_f^{(d)} | \vec{c}_f^{(d)}, \mathrm{MB}_{\mathcal{F}}(f)^{(d)})$$

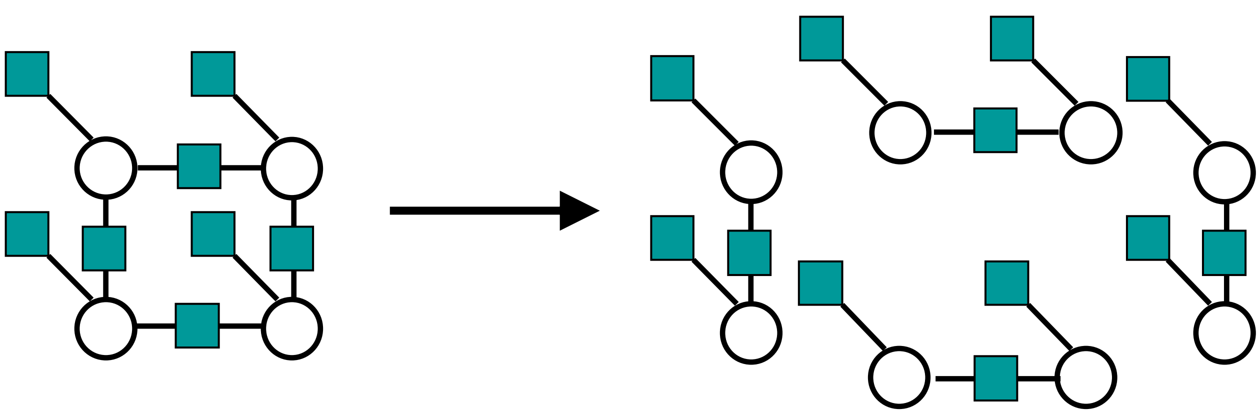
- Then inference during training is purely local
- Long range effects captured in data
- Note: only valid for training
  - in presence of fully labeled data

Local Training

- Break CRF into max likelihood trained pieces…
- Piecewise:



- Shared Unary Piecewise:



Evaluation

Models & Algorithms

- Model & algorithm specification:
  - Model / Training (/ Inference, if not obvious)
- Models & algorithms evaluated:
  - Indep / {Smallest, Largest} Pattern
  - Indep / BMA-Tree {Uniform, Exp}
  - Indep / Log Regr
  - CRF / ML Loopy BP (/ Swendsen-Wang)
  - Pattern CRF / Pseudolikelihood (Edge)
  - Pattern CRF / (S. U.) Piecewise
  - Monte Carlo

Training Time

- Approximate time for various models and algorithms to reach convergence:

Algorithm	**Training Time**
Indep / Largest Pattern	< 45 min
Indep / BMA-Tree	< 45 min
Pattern CRF / Piecewise	~ 2 hrs
Indep / Log Regr	~ 5 hrs
Pattern CRF / Pseudolikelihood	~ 12 hrs
CRF / ML Loopy BP	> 2 days

Inference Time

- Average time to evaluate $P(\vec{s}|\vec{c})$ for various models and algorithms on a 19x19 board:

Algorithm	**Inference Time**
Indep / Sm. & Largest Pattern	1.7 ms
Indep / BMA-Tree & Log Regr	6.0 ms
CRF / Loopy BP	101.0 ms
Pattern CRF / Loopy BP	214.6 ms
Monte Carlo	2,967.5 ms
CRF / Swend.-Wang Sampling	10,568.7 ms

Performance Metrics

- Vertex Error: (classification error)

$$\frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} \mathbb{I}(\mathrm{sgn}(\mathrm{E}_{P(\vec{s}|\vec{c}^{(d)})}[s_i]) \neq \mathrm{sgn}(s_i^{(d)}))$$

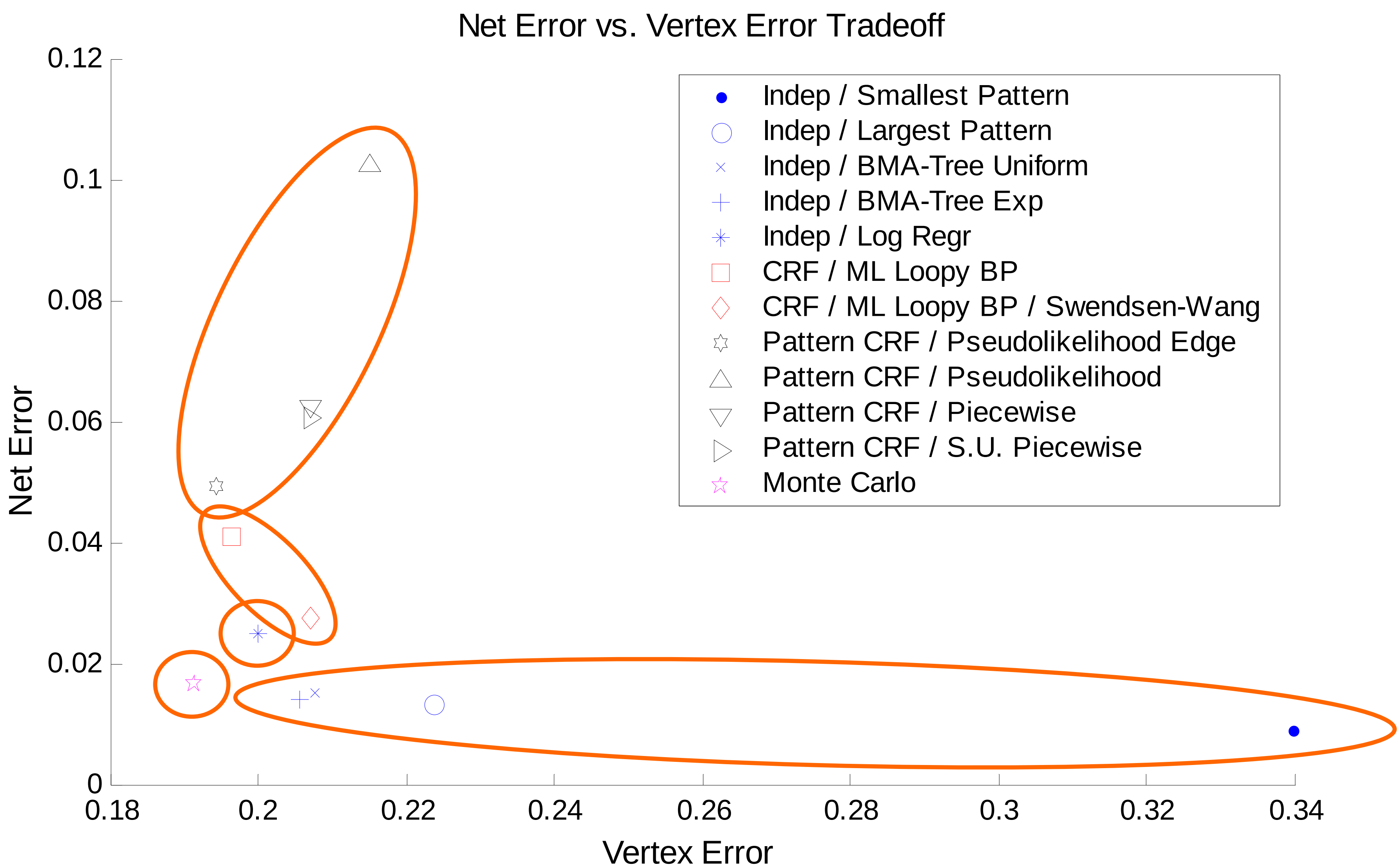
- Net Error: (score error)

$$\frac{1}{2|\mathcal{G}|} | \sum_{i=1}^{|\mathcal{G}|} \mathrm{E}_{P(\vec{s}|\vec{c}^{(d)})}[s_i] - \sum_{i=1}^{|\mathcal{G}|} s_i^{(d)} |$$

- Log Likelihood: (model fit)

$$\log P(\vec{s}^{(d)}|\vec{c}^{(d)})$$

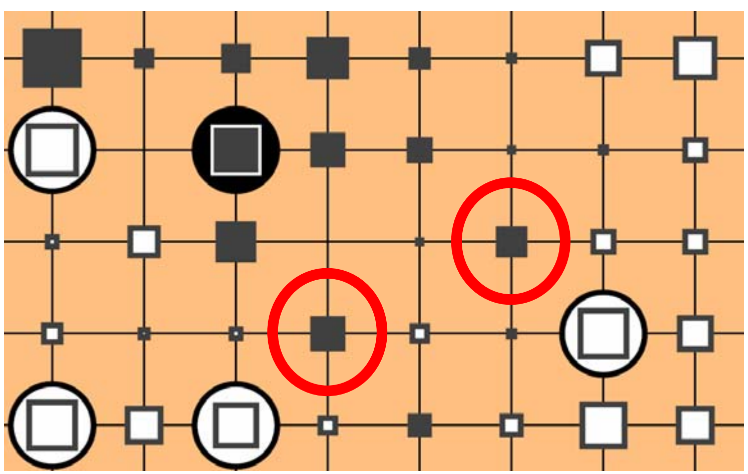
Performance Tradeoffs I



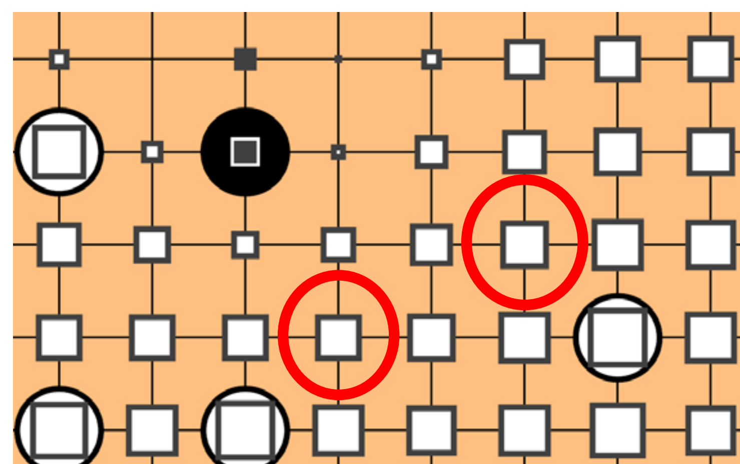
Why is Vertex Error better for CRFs?

- Coupling factors help realize stable configurations
- Compare previous unary-only independent model to unary and coupling model:
  - Independent models make inconsistent predictions
  - Loopy BP smoothes these predictions (but too much?)

BMA-Tree Model



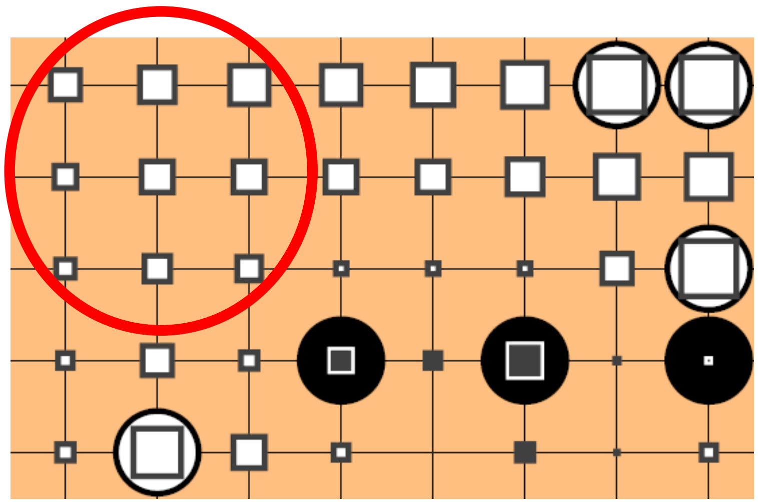
Coupling Model with Loopy BP



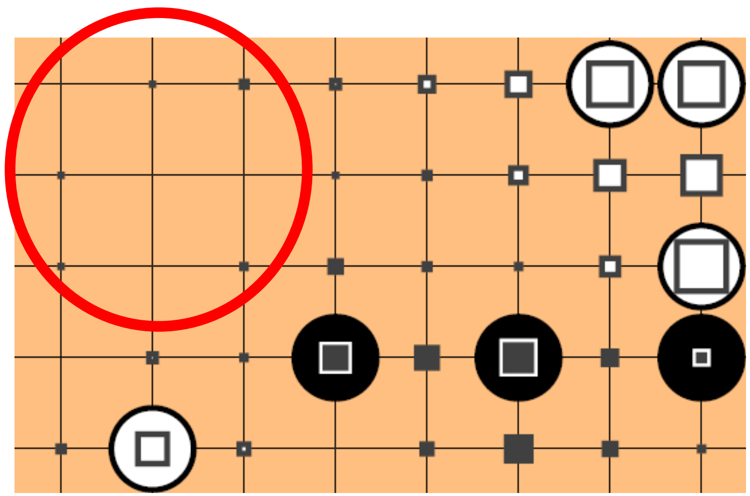
Why is Net Error worse for CRFs?

- Use sampling to examine bias of Loopy BP
  - Unbiased inference in limit
  - Can run over all test data but still too costly for training
- Smoothing gets rid of local inconsistencies
- But errors reinforce each other!

Loopy Belief Propagation



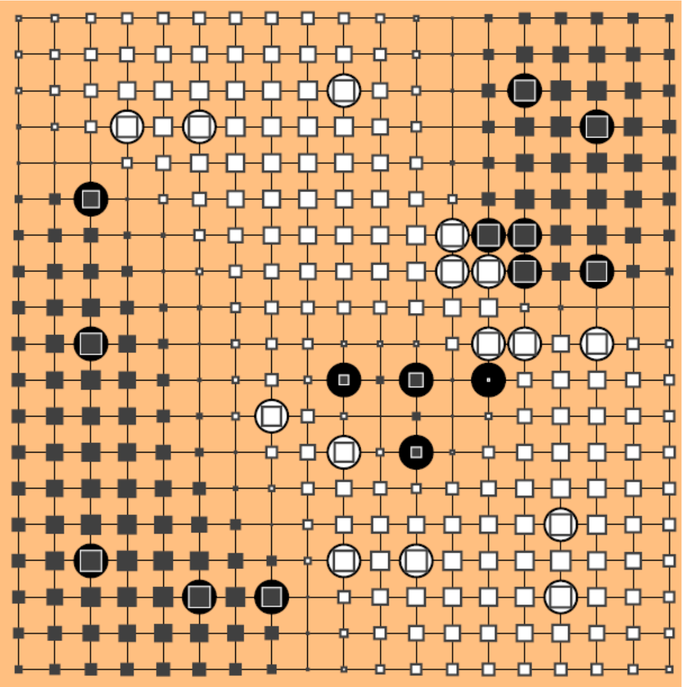
Swendsen-Wang Sampling



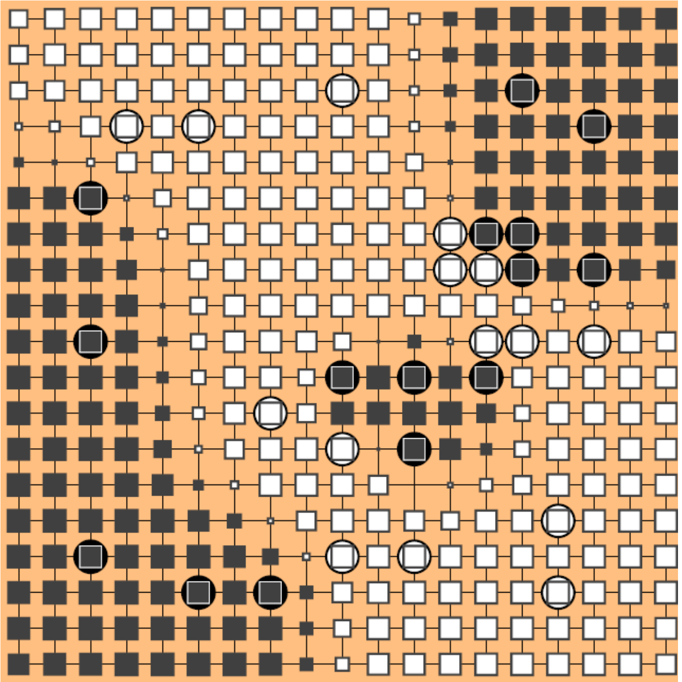
Bias of Local Training

- Problems with Piecewise training:
  – Very biased when used in conjunction with Loopy BP

ML Trained

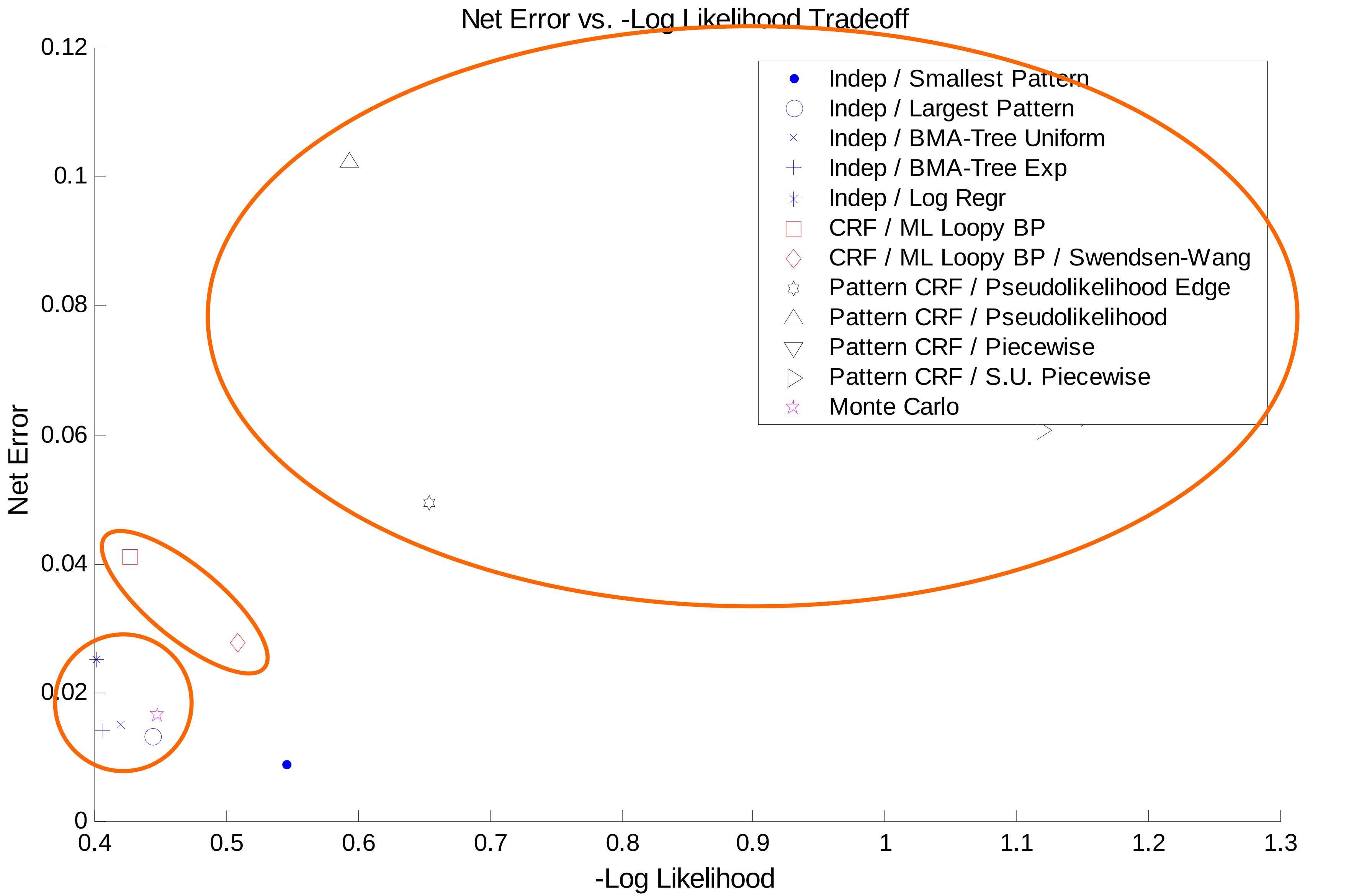


Piecewise Trained



  – Predictions still good, just saturated
  – Accounts for poor Log Likelihood & Net Error…

Performance Tradeoffs II



Conclusions

Two general messages:

(1) CRFs vs. Independent Models:

- CRFs should theoretically be better
- However, time cost is high
- Can save time with approximate training / inference
- But then CRFs may perform worse than independent classifiers – depends on metric

(2) For Independent Models:

- Problem of choosing appropriate neighborhood can be finessed by Bayesian model averaging