Soft Constraints:
Exponential Models

Factor graphs (undirected graphical models) and their connection to constraint programming
Soft constraint problems (*e.g.*, $MAX-SAT$)

- **Given**
  - n variables
  - m constraints, over various subsets of variables

- **Find**
  - Assignment to the n variables that maximizes the number of satisfied constraints.
Soft constraint problems (e.g, \textit{MAX-SAT})

- **Given**
  - n variables
  - m constraints, over various subsets of variables
  - m weights, one per constraint

- **Find**
  - Assignment to the n variables that maximizes the \textbf{total weight} of the satisfied constraints.
    - Equivalently, minimizes total weight of violated constraints.
Draw problem structure as a “factor graph”

Each constraint ("factor") is a function of the values of its variables.

- Measure goodness of an assignment by the **product of all the factors** ($\geq 0$).
  - How can we reduce previous slide to this?
    - There, each constraint was either satisfied or not (simple case).
    - There, good score meant large total weight for satisfied constraints.

figure thanks to Brian Potetz
Each constraint ("factor") is a function of the values of its variables.

- Measure goodness of an assignment by the \textbf{product of all the factors} (\(\geq 0\)).
  - How can we reduce previous slide to this?
    - There, each constraint was either satisfied or not (simple case).
    - There, good score meant \textbf{small} total weight for \textbf{violated} constraints.

\begin{itemize}
  \item Draw problem structure as a “factor graph”
  \item \textbf{variable}
  \item \textbf{factor graph}
  \item \textbf{measure goodness of an assignment by the product of all the factors (\(\geq 0\)).}
  \item \textbf{how can we reduce previous slide to this?}
  \item \textbf{there, each constraint was either satisfied or not (simple case).}
  \item \textbf{there, good score meant small total weight for violated constraints.}
\end{itemize}
Draw problem structure as a “factor graph”

Each constraint ("factor") is a function of the values of its variables.

- Measure goodness of an assignment by the **product of all the factors** ($\geq 0$).

- Models like this show up **all the time**.

---

figure thanks to Brian Potetz
Example: Ising Model
*(soft version of graph coloring, on a grid graph)*

<table>
<thead>
<tr>
<th>Model</th>
<th>Physics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean vars</td>
<td>Magnetic polarity at points on the plane</td>
</tr>
<tr>
<td>Binary equality constraints</td>
<td>?</td>
</tr>
<tr>
<td>Unary constraints</td>
<td>?</td>
</tr>
<tr>
<td>MAX-SAT</td>
<td>?</td>
</tr>
</tbody>
</table>

Figure thanks to ???
Example: Parts of speech
(or other sequence labeling problems)

```
Determiner  Noun  Aux  Adverb  Verb  Noun
```

```
this  can  can  really  can  tuna
```

Or, if the input words are given, you can customize the factors to them:

```
Determiner  Noun  Aux  Adverb  Verb  Noun
```

```
```
Local factors in a graphical model

First, a familiar example

- Conditional Random Field (CRF) for POS tagging

Possible tagging (i.e., assignment to remaining variables)

```
Observed input sentence (shaded)
```

```
find
preferred
tags
```
Local factors in a graphical model

First, a familiar example

- Conditional Random Field (CRF) for POS tagging

Possible tagging (i.e., assignment to remaining variables)
Another possible tagging

Observed input sentence (shaded)
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging

"Binary" factor that measures compatibility of 2 adjacent tags

Model reuses same parameters at this position
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging

“Unary” factor evaluates **this** tag
Its values depend on corresponding word

```
find
preferred
tag
  
  0.2
  0.2
  0

can’t be adj
```
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging

“Unary” factor evaluates this tag
Its values depend on corresponding word

(coULD BE MADE TO DEPEND ON entire observed sentence)
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging

“Unary” factor evaluates this tag
Different unary factor at each position

...  

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>n</td>
<td>a</td>
</tr>
<tr>
<td>0.3</td>
<td>0.02</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>n</td>
<td>a</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>n</td>
<td>a</td>
</tr>
<tr>
<td>0.2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

find  preferred  tags
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging

\[
p(v \ a \ n) \text{ is proportional to the product of all factors' values on } v \ a \ n
\]
Example: Medical diagnosis (QMR-DT)

- Patient is sneezing with a fever; no coughing

Diseases (about 600)

Symptoms (about 4000)

Patient is sneezing with a fever; no coughing
Example: Medical diagnosis

- Patient is sneezing with a fever; no coughing
  - Possible diagnosis: Flu (without coughing)
    - But maybe it’s not flu season …

Diseases

- Cold?
- Flu?
- Possessed?

Symptoms

- Sneezing?
- Fever?
- Coughing?
- Fits?
Example: Medical diagnosis

- Patient is sneezing with a fever; no coughing
  - Possible diagnosis: Cold (without coughing), and possessed (better ask about fits …)

```
Patient is sneezing with a fever; no coughing
- Possible diagnosis: Cold (without coughing), and possessed (better ask about fits …)
```

```
Diseases

Cold?  1
Flu?   0
Possessed?  1

Symptoms

Sneezing?  1
Fever?   1
Coughing?  0
Fits?    1

... ...
```

600.325/425 Declarative Methods - J. Eisner
Example: Medical diagnosis

- Patient is sneezing with a fever; no coughing
  - Possible diagnosis: Spontaneous sneezing, and possessed (better ask about fits …)

Note: Here symptoms & diseases are boolean. We could use real #s to denote degree.
Example: Medical diagnosis

- What are the factors, exactly?
- Factors that are \( w \) or 1 \( (\text{weighted MAX-SAT}) \):

According to whether some boolean constraint is true

If observe sneezing, get a disjunctive clause \((\text{Human} \lor \text{Cold} \lor \text{Flu})\)
If observe non-sneezing, get unit clauses \((\neg \text{Human}) \land (\neg \text{Cold}) \land (\neg \text{Flu})\)
Example: Medical diagnosis

- What are the factors, exactly?
- Factors that are probabilities:

\[
p(\text{Sneezing} \mid \text{Human, Cold, Flu})
\]

Use a little “noisy OR” model here:
\[
x = (\text{Human}, \text{Cold}, \text{Flu}), \text{ e.g., } (1,1,0).
\]
\[
p(\sim \text{sneezing} \mid x) = \exp(-w \cdot x)
\]

More 1’s should increase \( p(\text{sneezing}) \).
\[
e.g., w = (0.05, 2, 5)
\]

Would get logistic regression model if we replaced \( \exp \) by sigmoid, i.e., \( \exp/(1+\exp) \)
Example: Medical diagnosis

- What are the factors, exactly?
- Factors that are probabilities:
  - If observe sneezing, get a factor \((1 - \exp(- w \cdot x))\)
  - If observe non-sneezing, get a factor \(\exp(- w \cdot x)\)

As \(w \rightarrow \infty\), approach Boolean case (product of all factors \(\rightarrow 1\) if SAT, 0 if UNSAT)
Technique #1: Branch and bound

- Exact backtracking technique we’ve already studied.
  - And used via ECLiPSe’s “minimize” routine.
- Propagation can help prune branches of the search tree (add a hard constraint that we must do better than best solution so far).
- Worst-case exponential.

\[
\begin{array}{cccc}
(1,1,\ast) & (2,1,\ast) & (3,1,\ast) & (1,2,\ast) \\
(2,2,\ast) & (3,2,\ast) & (1,3,\ast) & (3,3,\ast) \\
(2,3,\ast) & (3,1,\ast) & (1,2,3) & (3,2,\ast) \\
(2,3,1) & (3,1,2) & (1,3,2) & (3,2,1) \\
\end{array}
\]
Technique #2: Variable Elimination

- Exact technique we’ve studied; worst-case exponential.

But how do we do it for soft constraints?

How do we join soft constraints?

Bucket E: \( E \neq D, \ E \neq C \)
Bucket D: \( D \neq A \)
Bucket C: \( C \neq B \)
Bucket B: \( B \neq A \)
Bucket A: contradiction

join all constraints in E’s bucket yielding a new constraint on D (and C)
now join all constraints in D’s bucket ...

figure thanks to Rina Dechter
Technique #2: Variable Elimination

- Easiest to explain via Dyna.

- goal max = f1(A,B) * f2(A,C) * f3(A,D) * f4(C,E) * f5(D,E).

- \text{tempE(C,D)} max = f4(C,E) * f5(D,E).

To eliminate E, join constraints mentioning E, and project E out.
Technique #2: Variable Elimination

- Easiest to explain via Dyna.

- goal \( \max = \text{tempD}(A,C) \times f_1(A,B) \times f_2(A,C) \times f_3(A,D) \times f_4(C,E) \times f_5(D,E) \).

- \( \text{tempD}(A,C) \max = \text{tempE}(C,D) \times f_3(A,D) \).

- to eliminate \( D \), join constraints mentioning \( D \), and project \( D \) out.
Technique #2: Variable Elimination

- Easiest to explain via Dyna.

- goal max = f1(A,B)*f2(A,C)*tempD(A,C).
  tempC(A)

- tempC(A) max = f2(A,C)*tempD(A,C).

- tempD(A,C) max = f3(A,D)*tempE(C,D).

- tempE(C,D) max = f4(C,E)*f5(D,E).
Technique #2: Variable Elimination

- Easiest to explain via Dyna.

- goal max= tempC(A)*f1(A,B). tempB(A)
- tempB(A) max= f1(A,B).
- tempC(A) max= f2(A,C)*tempD(A,C).
- tempD(A,C) max= f3(A,D)*tempE(C,D).
- tempE(C,D) max= f4(C,E)*f5(D,E).
Technique #2: Variable Elimination

- Easiest to explain via Dyna.

- goal \(\text{max} = \text{tempC}(A) \times \text{tempB}(A)\).
- \(\text{tempB}(A)\) \(\text{max} = f_1(A,B)\).
- \(\text{tempC}(A)\) \(\text{max} = f_2(A,C) \times \text{tempD}(A,C)\).
- \(\text{tempD}(A,C)\) \(\text{max} = f_3(A,D) \times \text{tempE}(C,D)\).
- \(\text{tempE}(C,D)\) \(\text{max} = f_4(C,E) \times f_5(D,E)\).
Probabilistic interpretation of factor graph
("undirected graphical model")

Each factor is a function \( \geq 0 \) of the values of its variables.

- For any assignment \( x = (x_1, \ldots, x_5) \), define \( u(x) = \text{product of all factors} \), e.g.,
  \[ u(x) = f_1(x) * f_2(x) * f_3(x) * f_4(x) * f_5(x). \]
- We’d like to interpret \( u(x) \) as a probability distribution over all \( 2^5 \) assignments.

- Do we have \( u(x) \geq 0? \) Yes. ☺
- Do we have \( \sum u(x) = 1? \)
  No. \( \sum u(x) = Z \) for some \( Z \). ☹
- So \( u(x) \) is not a probability distribution.
- But \( p(x) = u(x)/Z \) is!
Z is hard to find … (the “partition function”)

- Exponential time with this Dyna program.

- goal $\max = f_1(A, B) \times f_2(A, C) \times f_3(A, D) \times f_4(C, E) \times f_5(D, E)$.

  $\quad + =$

  This explicitly sums over all $2^5$ assignments.
  We can do better by variable elimination ...
  (although still exponential time in worst case).
  Same algorithm as before: just replace $\max = \ $ with $+ =$. 
Z is hard to find … (the “partition function”)

- Faster version of Dyna program, after var elim.

- goal += tempC(A)*tempB(A).
- tempB(A) += f1(A,B).
- tempC(A) += f2(A,C)*tempD(A,C).
- tempD(A,C) += f3(A,D)*tempE(C,D).
- tempE(C,D) += f4(C,E)*f5(D,E).
Why a probabilistic interpretation?

1. Allows us to make predictions.
   - You're sneezing with a fever & no cough.
   - Then what is the probability that you have a cold?

2. Important in learning the factor functions.
   - Maximize the probability of training data.

3. Central to deriving fast approximation algorithms.
   - “Message passing” algorithms where nodes in the factor graph are repeatedly updated based on adjacent nodes.
   - Many such algorithms. E.g., survey propagation is the current best method for random 3-SAT problems. Hot area of research!
Probabilistic interpretation ➔ Predictions

You’re sneezing with a fever & no cough.

Then what is the *probability* that you have a cold?

- Randomly sample 10000 assignments from \( p(x) \).
- In 200 of them (2%), patient is sneezing with a fever and no cough.
- In 140 (1.4%) of those, the patient *also* has a cold.

\[ \text{answer: 70\% (140/200)} \]
Probabilistic interpretation ➔ Predictions

You’re sneezing with a fever & no cough.

Then what is the *probability* that you have a cold?

- Randomly sample 10000 assignments from p(x).
- In 200 of them (2%), patient is sneezing with a fever and no cough.
- In 140 (1.4%) of those, the patient *also* has a cold.

All samples

<table>
<thead>
<tr>
<th>sneezing, fever, etc.</th>
<th>p=0.02</th>
</tr>
</thead>
<tbody>
<tr>
<td>also a cold</td>
<td>p=0.014</td>
</tr>
</tbody>
</table>

answer: 70% \( \frac{0.014}{0.02} \)
Probabilistic interpretation ➔ Predictions

You’re sneezing with a fever & no cough. Then what is the **probability** that you have a cold?

- Randomly sample 10000 assignments from \( p(x) \).
- In 200 of them (2%), patient is sneezing with a fever and no cough.
- In 140 (1.4%) of those, the patient also has a cold.

answer: 70% \( \frac{0.014 \cdot Z}{0.02 \cdot Z} \)
You’re sneezing with a fever & no cough.
Then what is the probability that you have a cold?

- Randomly sample 10000 assignments from p(x).

Remember, we can find this by variable elimination.
This too: just add unary constraints Sneezing=1,Fever=1,Cough=0
This too: one more unary constraint Cold=1

Could we compute exactly instead? 
unnecessary

all samples
sneezing, fever, etc. $u = 0.02 \cdot Z$
also a cold $u = 0.014 \cdot Z$

answer: 70% ($0.014 \cdot Z / 0.02 \cdot Z$)
Probabilistic interpretation ➔ Learning

- How likely is it for \((X_1, X_2, X_3) = (1,0,1)\) (according to real data)? 90% of the time.
- How likely is it for \((X_1, X_2, X_3) = (1,0,1)\) (according to the full model)? 55% of the time.
  - I.e., if you randomly sample many assignments from \(p(x)\), 55% of assignments have (1,0,1).
  - E.g., 55% have (Cold, ~Cough, Sneeze): too few.
- To learn a better \(p(x)\), we adjust the factor functions to bring the second ratio from 55% up to 90%.
Probabilistic interpretation ➔ Learning

- How likely is it for \((X_1, X_2, X_3) = (1, 0, 1)\) (according to real data)? 90% of the time
- How likely is it for \((X_1, X_2, X_3) = (1, 0, 1)\) (according to the full model)? 55% of the time
- To learn a better \(p(x)\), we adjust the factor functions to bring the second ratio from 55% up to 90%.

- By increasing \(f_1(1, 0, 1)\), we can increase the model’s probability that \((X_1, X_2, X_3) = (1, 0, 1)\).
- Unwanted ripple effect: This will also increase the model’s probability that \(X_3=1\), and hence will change the probability that \(X_5=1\), and ...
- So we have to change all the factor functions at once to make all of them match real data.
- Theorem: This is always possible. (gradient descent or other algorithms)
  - Theorem: The resulting learned function \(p(x)\) maximizes \(p(\text{real data})\).
Probabilistic interpretation ➔ Learning

- How likely is it for \((X_1, X_2, X_3) = (1, 0, 1)\) (according to real data)? 90% of the time
- How likely is it for \((X_1, X_2, X_3) = (1, 0, 1)\) (according to the full model)? 55% of the time
- To learn a better \(p(x)\), we adjust the factor functions to bring the second ratio from 55% up to 90%.

- By increasing \(f_1(1, 0, 1)\), we can increase the model’s probability that \((X_1, X_2, X_3) = (1, 0, 1)\).
- Unwanted ripple effect: This will also increase the model’s probability that \(X_3 = 1\), and hence will change the probability that \(X_5 = 1\), and …
- So we have to change all the factor functions at once to make all of them match real data.
- Theorem: This is always possible. (gradient descent or other algorithms)
  - Theorem: The resulting learned function \(p(x)\) maximizes \(p(\text{real data})\).
Probabilistic interpretation → Approximate constraint satisfaction

3. Central to deriving fast **approximation** algorithms.
   - “Message passing” algorithms where nodes in the factor graph are repeatedly updated based on adjacent nodes.

- Gibbs sampling / simulated annealing
- Mean-field approximation and other variational methods
- Belief propagation
- Survey propagation
How do we sample from $p(x)$?

- **Gibbs sampler:** *(should remind you of stochastic SAT solvers)*
  - Pick a random starting assignment.
  - Repeat $n$ times: Pick a variable and possibly flip it, at random
  - **Theorem:** Our new assignment is a random sample from a distribution close to $p(x)$ *(converges to $p(x)$ as $n \to \infty$)*

How do we decide whether new value should be 0 or 1?

If $u(x)$ is twice as big when set at 0 than at 1, then pick 1 with prob $2/3$, pick 0 with prob $1/3$.

It’s a local computation to determine that flipping the variable doubles $u(x)$, since only these factors of $u(x)$ change.
Technique #3: Simulated annealing

- Gibbs sampler can sample from $p(x)$.
- Replace each factor $f(x)$ with $f(x)\beta$.
- Now $p(x)$ is proportional to $u(x)\beta$, with $\sum p(x) = 1$.
- What happens as $\beta \to \infty$?

- Sampler turns into a maximizer!
  - Let $x^*$ be the value of $x$ that maximizes $p(x)$.
  - For very large $\beta$, a single sample is almost always equal to $x^*$.

- Why doesn’t this mean $P=NP$?
  - As $\beta \to \infty$, need to let $n \to \infty$ too to preserve quality of approx.
    - Sampler rarely goes down steep hills, so stays in local maxima for ages.
  - Hence, simulated annealing: gradually increase $\beta$ as we flip variables.
  - Early on, we’re flipping quite freely
Technique #4: Variational methods

- To work exactly with $p(x)$, we’d need to compute quantities like $Z$, which is NP-hard.
  - (e.g., to predict whether you have a cold, or to learn the factor functions)
- We saw that Gibbs sampling was a good (but slow) approximation that didn’t require $Z$.
- The mean-field approximation is sort of like a deterministic “averaged” version of Gibbs sampling.
  - In Gibbs sampling, nodes flutter on and off – you can ask how often $x_3$ was 1.
  - In mean-field approximation, every node maintains a belief about how often it’s 1. This belief is updated based on the beliefs at adjacent nodes. No randomness.
  - [details beyond the scope of this course, but within reach]
Technique #4: Variational methods

- The mean-field approximation is sort of like a deterministic “averaged” version of Gibbs sampling.
  - In Gibbs sampling, nodes flutter on and off – you can ask how often $x_3$ was 1.
  - In mean-field approximation, every node maintains a belief about how often it’s 1. This belief is repeatedly updated based on the beliefs at adjacent nodes. No randomness.

![Diagram](attachment:image.png)

Set this now to 0.6
Technique #4: Variational methods

- The mean-field approximation is sort of like a deterministic “averaged” version of Gibbs sampling.
  - Can frame this as seeking an optimal approximation of this $p(x)$ ... 
  - ... by a $p(x)$ defined as a product of simpler factors (easy to work with):
Technique #4: Variational methods

More sophisticated version: Belief Propagation

- The soft version of arc consistency
  - Arc consistency: some of my values become impossible $\Rightarrow$ so do some of yours
  - Belief propagation: some of my values become unlikely $\Rightarrow$ so do some of yours
    - Therefore, your other values become more likely

Note: Belief propagation has to be more careful than arc consistency about not having X’s influence on Y feed back and influence X as if it were separate evidence. Consider constraint $X=Y$.

- But there will be feedback when there are cycles in the factor graph – which hopefully are long enough that the influence is not great. If no cycles (a tree), then the beliefs are exactly correct. In this case, BP boils down to a dynamic programming algorithm on the tree.

Can also regard it as Gibbs sampling without the randomness

- That’s what we said about mean-field, too, but this is an even better approx.
- Gibbs sampling lets you see:
  - how often $x_1$ takes each of its 2 values, 0 and 1.
  - how often $(x_1,x_2,x_3)$ takes each of its 8 values such as $(1,0,1)$.
    (This is needed in learning if $(x_1,x_2,x_3)$ is a factor.)
- Belief propagation estimates these probabilities by “message passing.”
- Let’s see how it works!
Technique #4: Variational methods

- Mean-field approximation
- Belief propagation
- Survey propagation:
  - Like belief propagation, but also assess the belief that the value of this variable doesn’t matter! Useful for solving hard random 3-SAT problems.
- Generalized belief propagation: Joins constraints, roughly speaking.
- Expectation propagation: More approximation when belief propagation runs too slowly.
- Tree-rewighted belief propagation: …
Great Ideas in ML: Message Passing

Count the soldiers

there’s 1 of me

1 before you
2 before you
3 before you
4 before you
5 before you

5 behind you
4 behind you
3 behind you
2 behind you
1 behind you

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Count the soldiers

Belief: Must be 2 + 1 + 3 = 6 of us

there's 1 of me

2 before you

only see my incoming messages

3 behind you

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Count the soldiers

Belief: Must be 1 + 1 + 4 = 6 of us
Belief: Must be 1 + 1 + 3 = 6 of us

there's 1 of me
1 before you
only see my incoming messages
4 behind you

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of the tree

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

Belief: Must be 14 of us

 wouldn’t work correctly with a “loopy” (cyclic) graph

adapted from MacKay (2003) textbook
In the CRF, message passing = forward-backward
Great ideas in ML: **Loopy Belief Propagation**

- Extend CRF to “skip chain” to capture non-local factor
  - More influences on belief

![Diagram of Loopy Belief Propagation]

Great ideas in ML:

- Loopy Belief Propagation

![Diagram of Loopy Belief Propagation]

- ... find preferred tags ...
Great ideas in ML: **Loopy Belief Propagation**

- Extend CRF to “skip chain” to capture non-local factor
  - More influences on belief
  - Graph becomes loopy 🤔

Red messages not independent? Pretend they are!

Great ideas in ML:

- Loopy Belief Propagation
Technique #4: Variational methods

- Mean-field approximation
- Belief propagation
- Survey propagation:
  - Like belief propagation, but also assess the belief that the value of this variable doesn’t matter! Useful for solving hard random 3-SAT problems.
- Generalized belief propagation: Joins constraints, roughly speaking.
- Expectation propagation: More approximation when belief propagation runs too slowly.
- Tree-reweighted belief propagation: …