Predicting Sequences: Global Models

CS 6355: Structured Prediction

THE UNIVERSITY OF UTAH

Outline

- Sequence models
- Hidden Markov models
  - Inference with HMM
  - Learning
- Conditional Models and Local Classifiers
- Global models
  - *Conditional* Random Fields
  - Structured Perceptron for sequences

So far…

- Hidden Markov models
  - **Pros**: Decomposition of total probability with tractable inference
  - **Cons**: Doesn't allow use of features for representing inputs
    - Also, generative model

      (not really a downside, but we may get better performance with conditional models if we care only about predictions)

- Local, conditional Markov Models
  - **Pros**: Conditional model, allows features to be used, tractable inference
  - **Cons**: Label bias problem

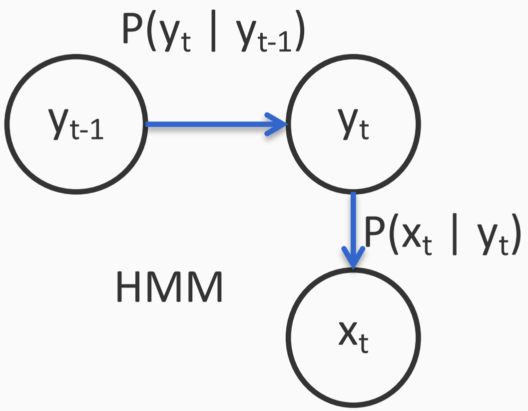
Global models

- Train the predictor globally
  - Instead of training local decisions independently

- *Normalize globally*
  - Make each edge in the model undirected
  - Not associated with a probability, but just a “score”

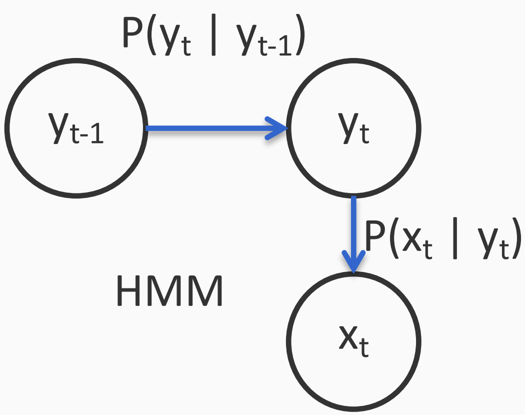
- Recall the difference between local vs. global for multiclass

HMM vs. A local model vs. A global model

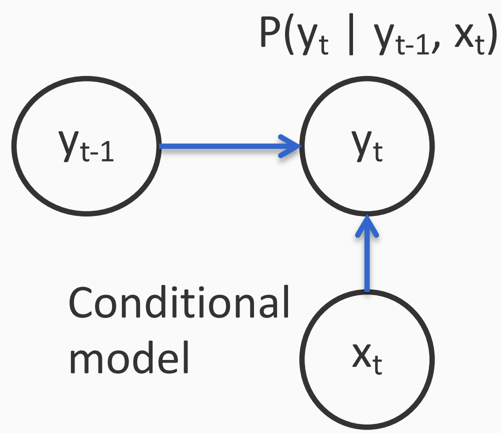


Generative

HMM vs. *A local model* vs. A global model



HMM



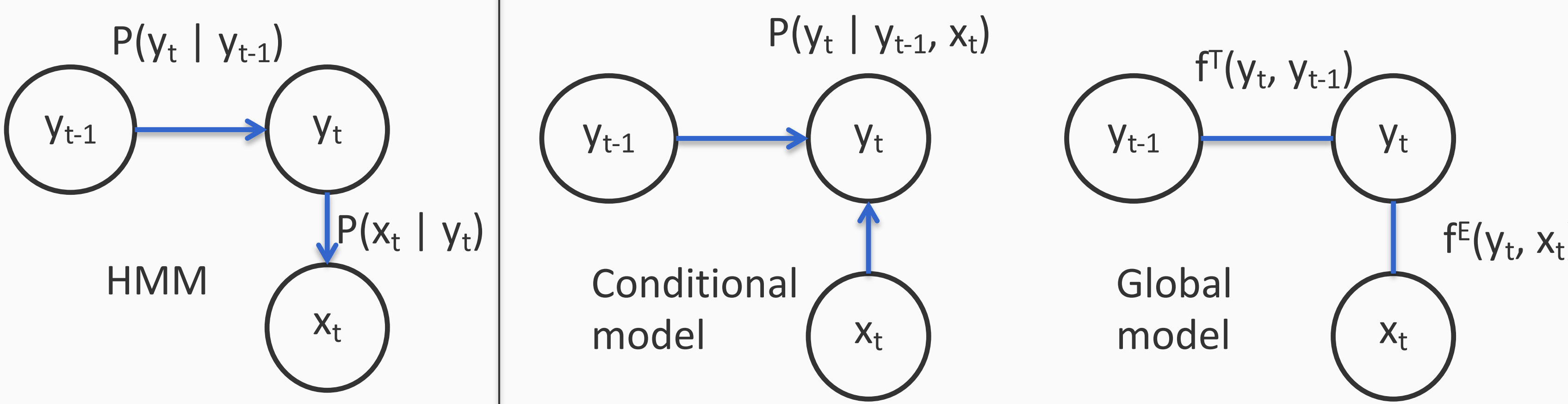
Conditional model

Generative

Discriminative

Local: P is locally normalized to add up to one for each step

HMM vs. A local model vs. *A global model*



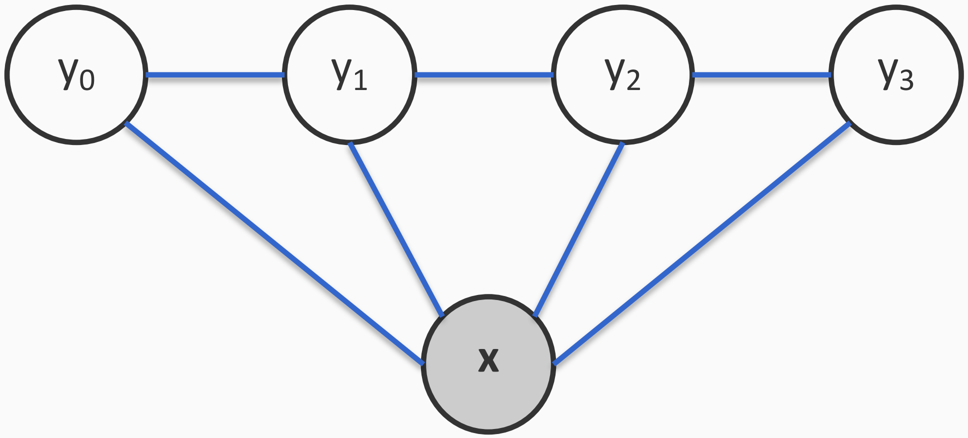
Generative

Discriminative

Local: P is locally normalized to add up to one for each step

Global: The functions f^T and f^E are scores that are not normalized

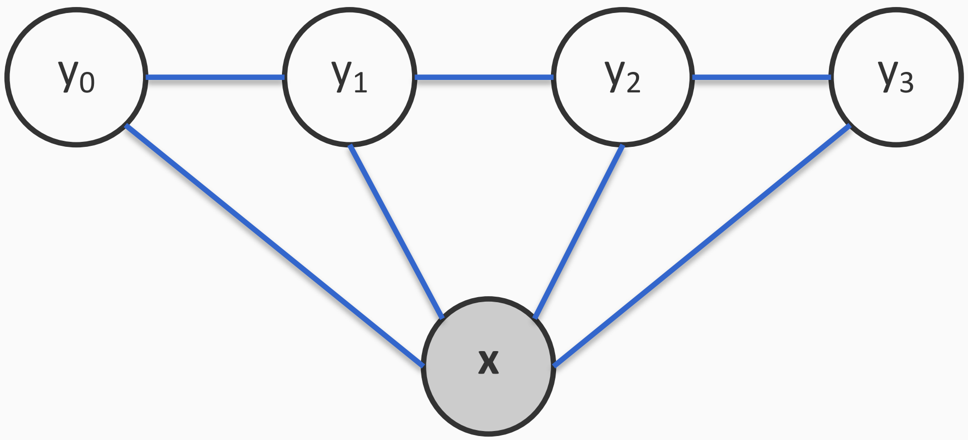
Conditional Random Field



Each node is a random variable

We observe some nodes and the rest are unobserved

Conditional Random Field



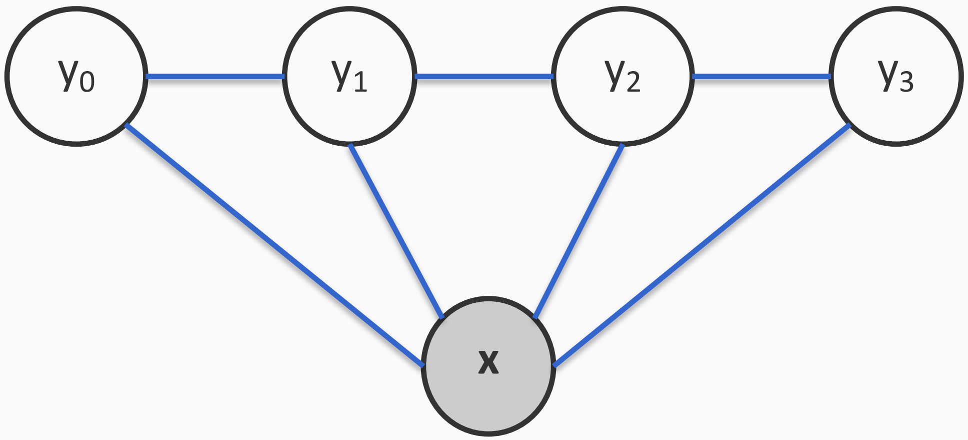
Each node is a random variable

We observe some nodes and the rest are unobserved

For example:

- x could be a random variable representing an input video,
- the y's could represent whether the corresponding time step is at the start, end, or within a scene.

Conditional Random Field



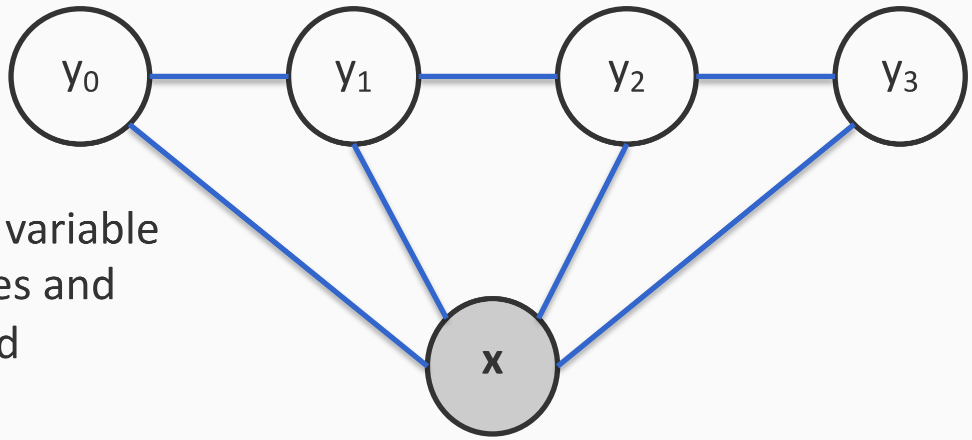
Each node is a random variable

We observe some nodes and the rest are unobserved

The goal: To characterize a probability distribution over the unobserved variables, conditioned on the observed ones.

That is, to characterize $P(\, y_0, y_1, \cdots \mid \mathbf{x}\,)$.

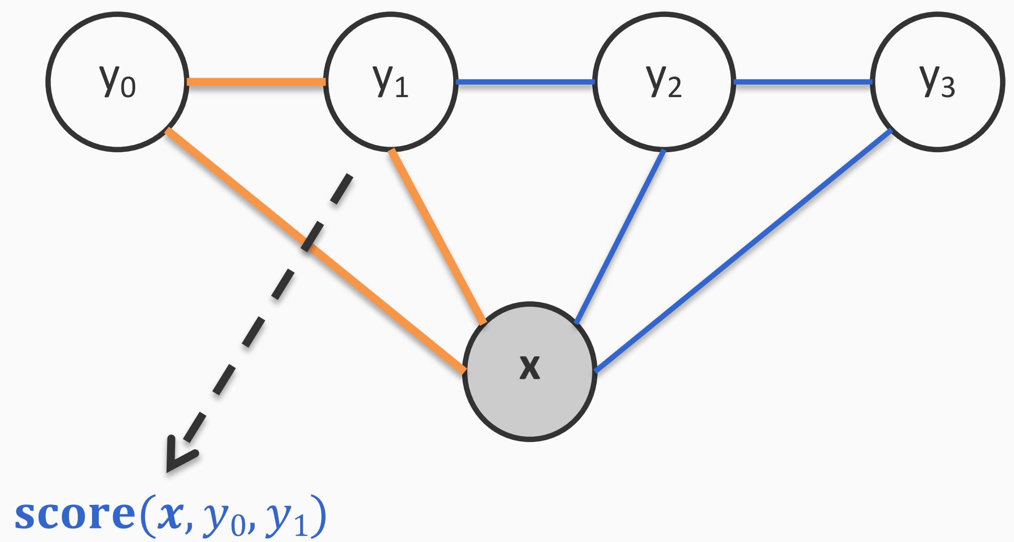
Scoring assignments to outputs



Each node is a random variable
We observe some nodes and
the rest are unobserved

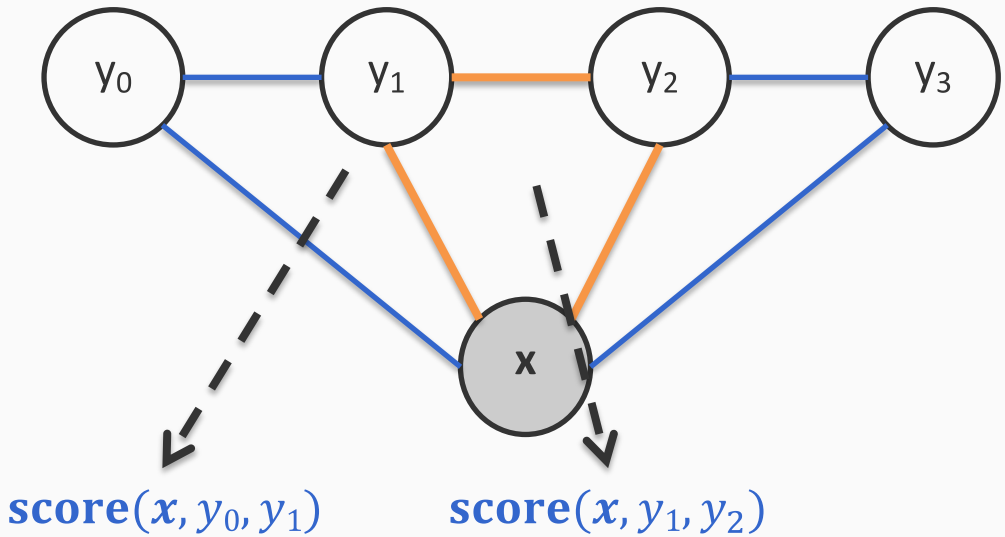
The strategy: Each ***clique*** is associated with a score

Scoring assignments to outputs



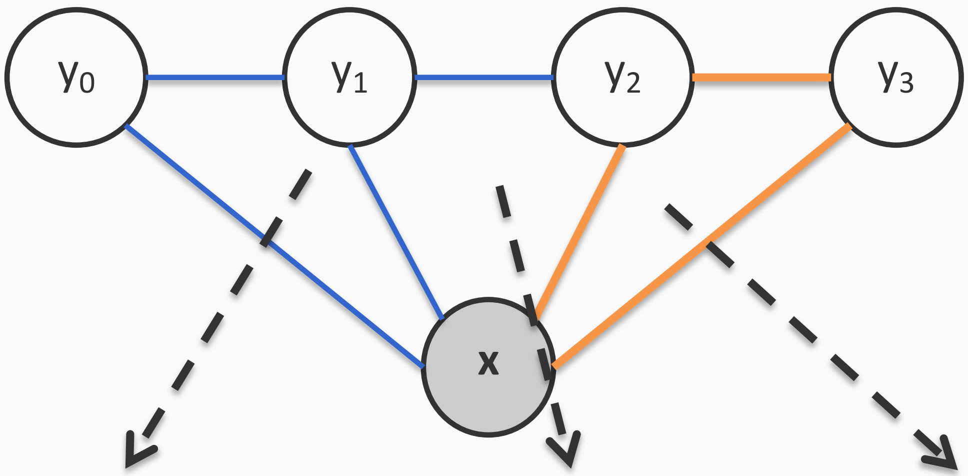
The strategy: Each ***clique*** is associated with a score

Scoring assignments to outputs



The strategy: Each ***clique*** is associated with a score

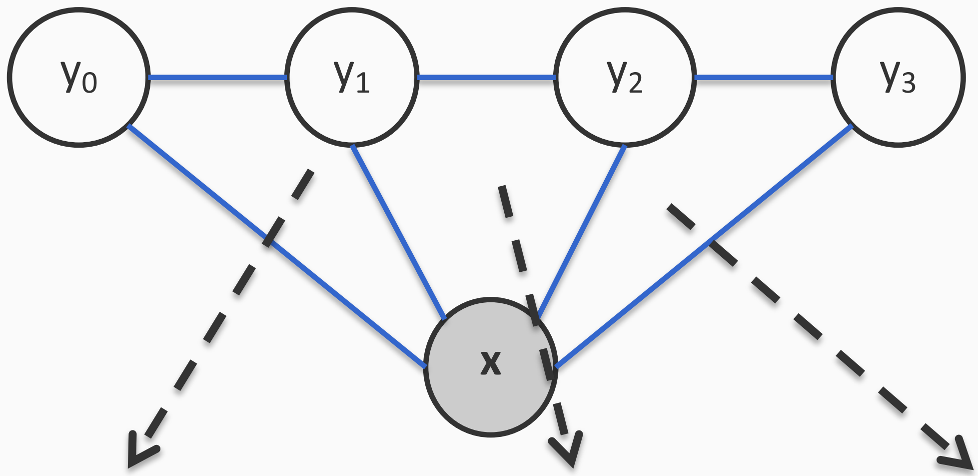
Scoring assignments to outputs



$\mathbf{score}(\boldsymbol{x}, y_0, y_1)$ $\mathbf{score}(\boldsymbol{x}, y_1, y_2)$ $\mathbf{score}(\boldsymbol{x}, y_2, y_3)$

The strategy: Each ***clique*** is associated with a score

Scoring assignments to outputs



$$\mathbf{score}(\boldsymbol{x}, y_0, y_1) = \mathbf{w}^T \phi(x, y_0, y_1)$$

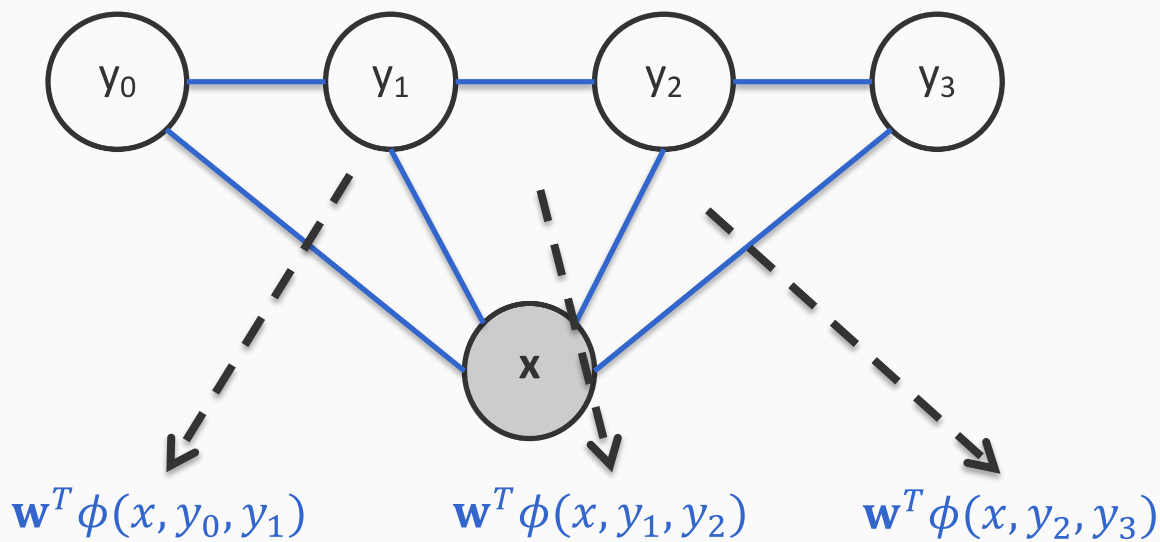
$$\mathbf{score}(\boldsymbol{x}, y_1, y_2) = \mathbf{w}^T \phi(x, y_1, y_2)$$

$$\mathbf{score}(\boldsymbol{x}, y_2, y_3) = \mathbf{w}^T \phi(x, y_2, y_3)$$

The strategy: Each ***clique*** is associated with a score

The usual scoring function: A linear function of weights and features of the associated nodes

Scoring assignments to outputs



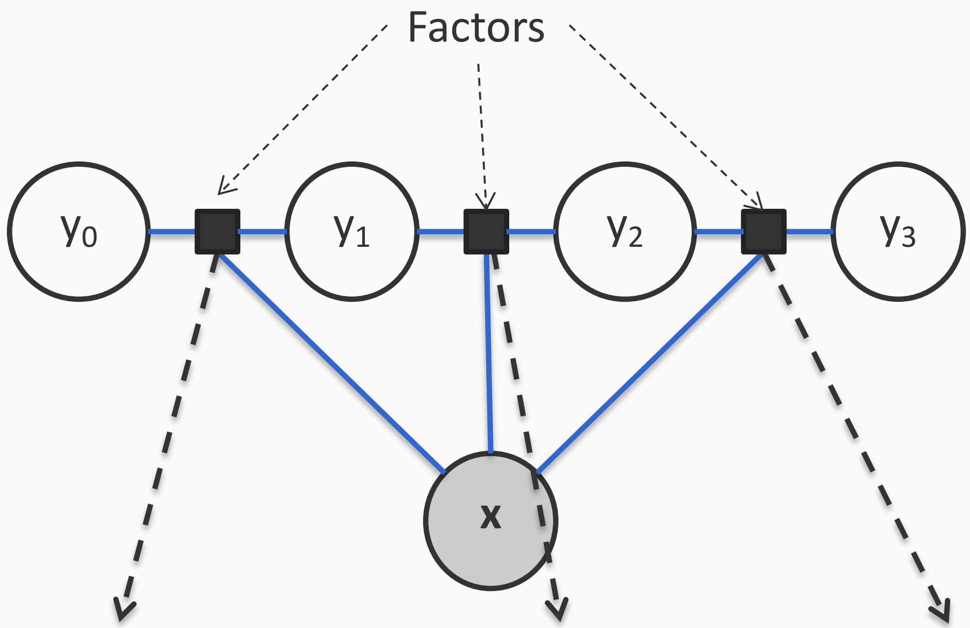
Each node is a random variable

We observe some nodes and need to assign the rest

Each ***clique*** is associated with a score, typically linear

Arbitrary features, as with local conditional models

Another notation: A *factor graph*



$\boldsymbol{w}^T\phi(\boldsymbol{x}, y_0, y_1)$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_1, y_2)$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_2, y_3)$

Each node is a random variable

We observe some nodes and need to assign the rest

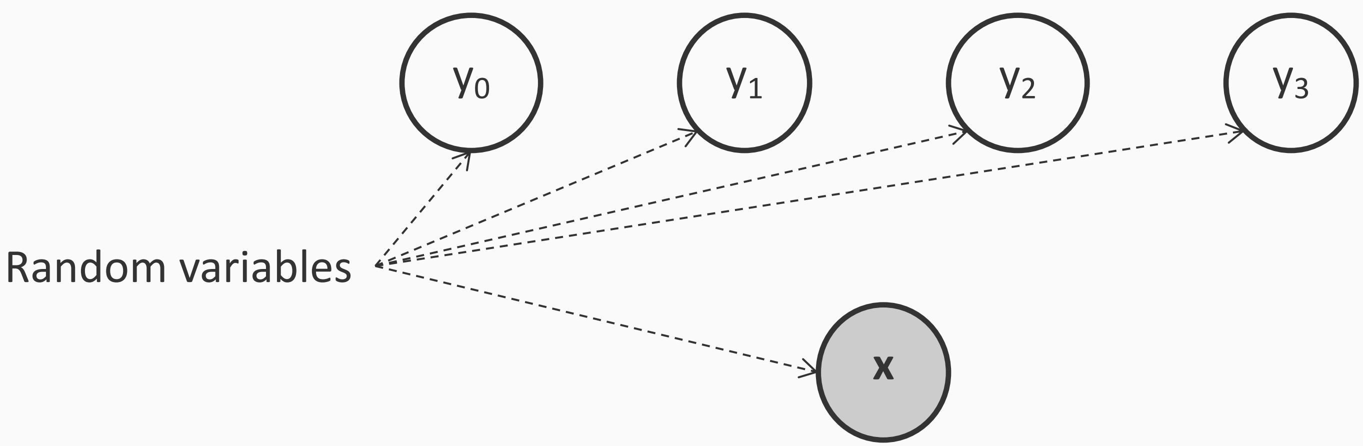
Each ~~clique~~ factor is associated with a score

Notation: A factor graph

- A bipartite graph consisting of two kinds of nodes

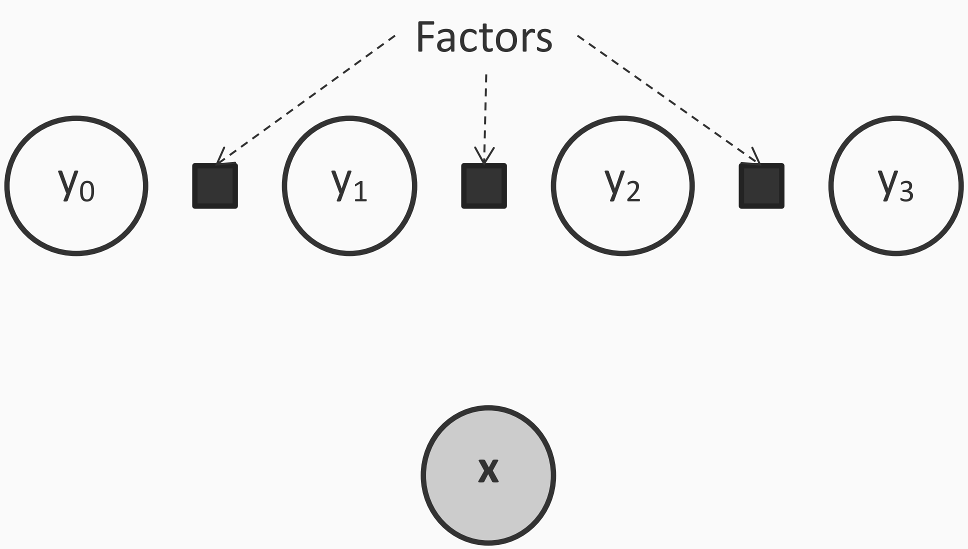
Notation: A factor graph

- A bipartite graph consisting of two kinds of nodes
  - Random variables (usually circles) represent decisions



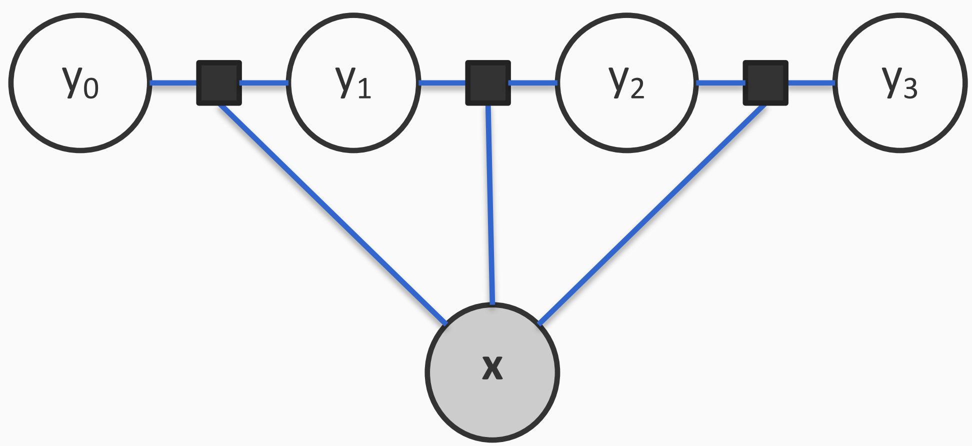
Notation: A factor graph

- A bipartite graph consisting of two kinds of nodes
  - Random variables (usually circles) represent decisions
  - Factors (usually squares) represent interactions



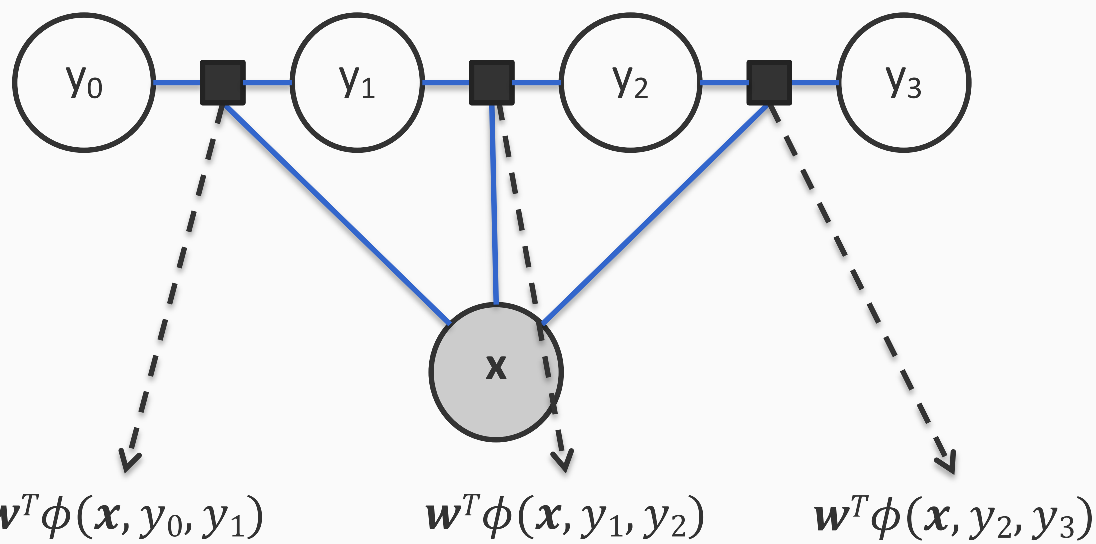
Notation: A factor graph

- A bipartite graph consisting of two kinds of nodes
  - Random variables (usually circles) represent decisions
  - Factors (usually squares) represent interactions
  - Edges: Random variables that interact with each other (think parts)

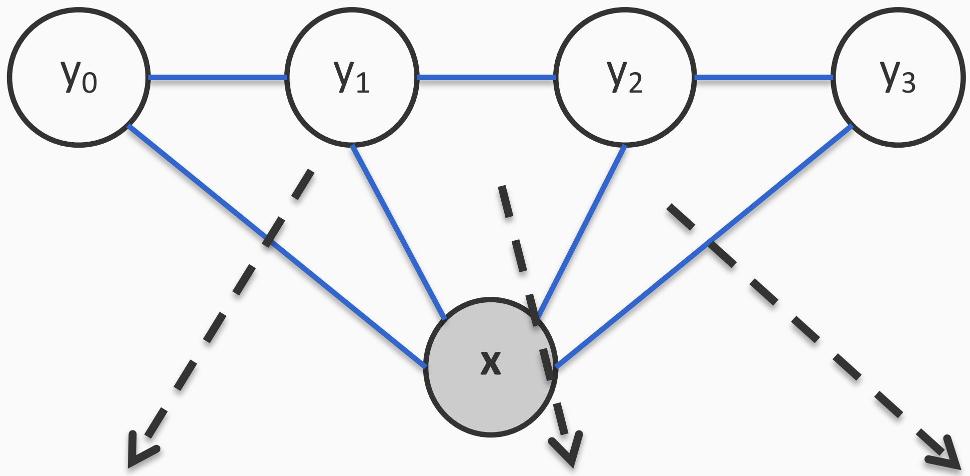


Notation: A factor graph

- A bipartite graph consisting of two kinds of nodes
  - Random variables (usually circles) represent decisions
  - Factors (usually squares) represent interactions
- Semantics: All random variables that are connected to a factor are scored together. *That is, each factor corresponds to a score.*



Scoring assignments to outputs



$\mathbf{w}^T \phi(x, y_0, y_1)$ $\mathbf{w}^T \phi(x, y_1, y_2)$ $\mathbf{w}^T \phi(x, y_2, y_3)$

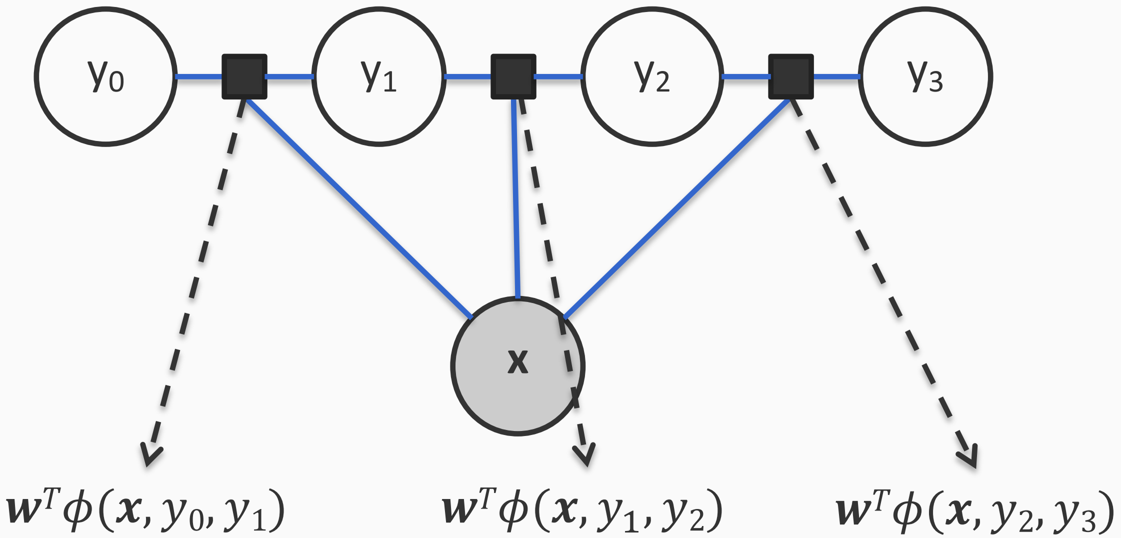
Each node is a random variable

We observe some nodes and need to assign the rest

Each ***clique*** is associated with a score, typically linear

Arbitrary features, as with local conditional models

Scoring with factor graphs



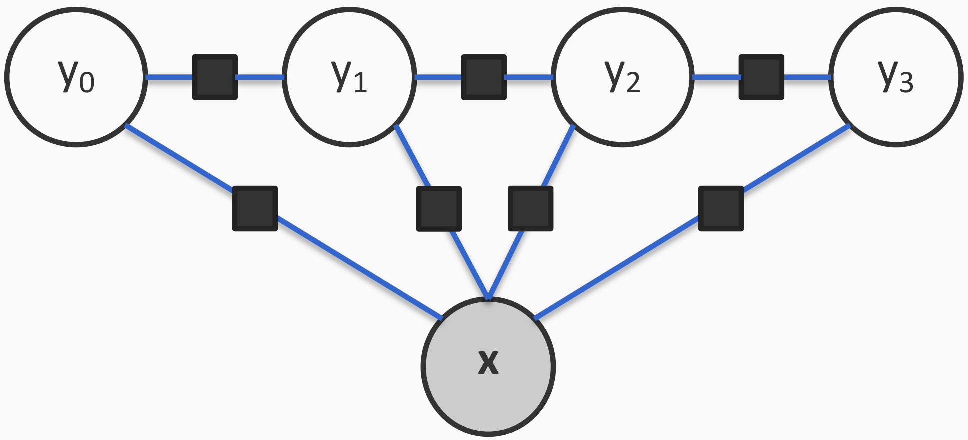
Each node is a random variable

We observe some nodes and need to assign the rest

Each ~~clique~~ factor is associated with a score

Scoring with factor graphs

A different factorization: Recall decomposition of structures into parts. Same idea



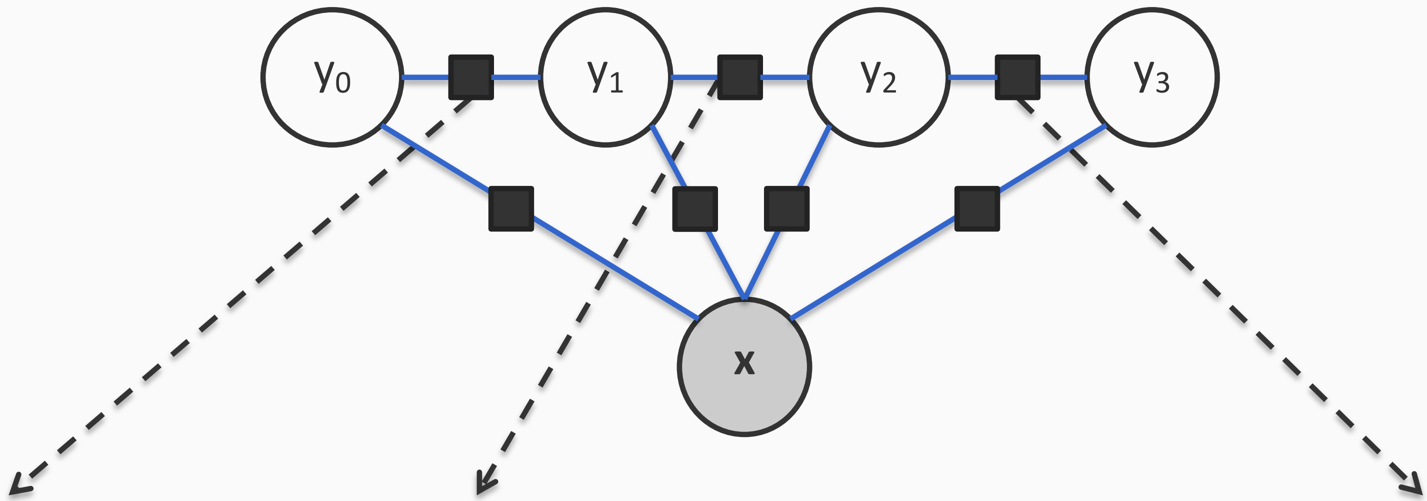
Each node is a random variable

We observe some nodes and need to assign the rest

Each factor is associated with a score

Scoring with factor graphs

A different factorization: Recall decomposition of structures into parts. Same idea



$\mathbf{w}^T\phi(y_0, y_1)$ $\mathbf{w}^T\phi(y_1, y_2)$ $\mathbf{w}^T\phi(y_2, y_3)$

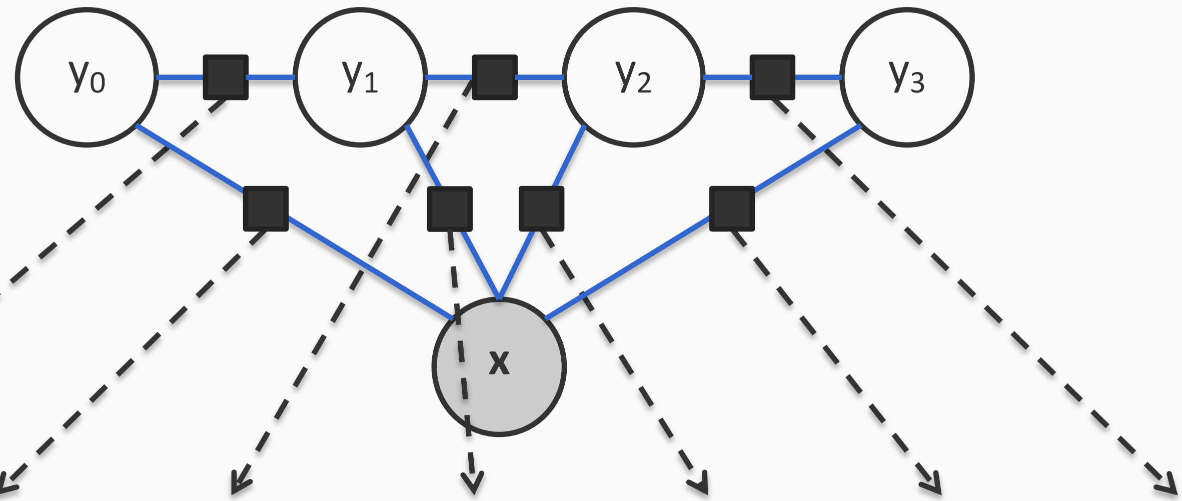
Each node is a random variable

We observe some nodes and need to assign the rest

Each factor is associated with a score

Scoring with factor graphs

A different factorization: Recall decomposition of structures into parts. Same idea



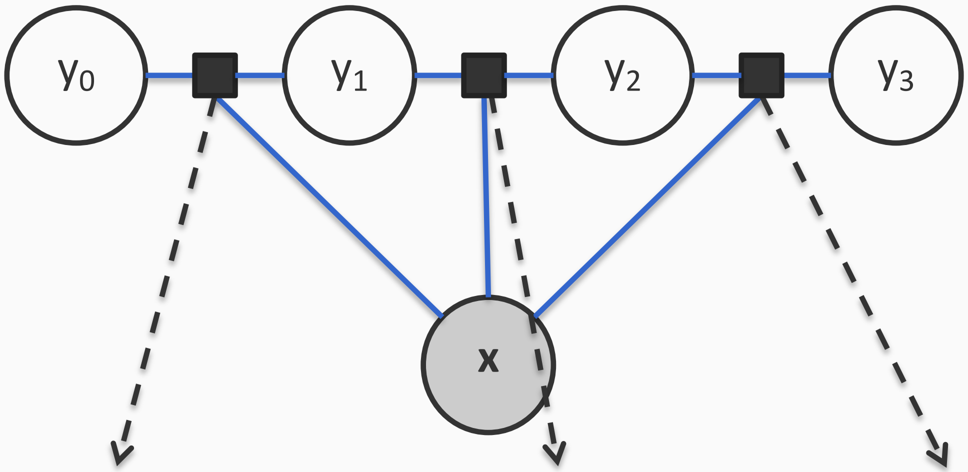
$\mathbf{w}^T\phi(y_0, y_1)$ $\mathbf{w}^T\phi(y_0, \mathbf{x})$ $\mathbf{w}^T\phi(y_1, y_2)$ $\mathbf{w}^T\phi(y_1, \mathbf{x})$ $\mathbf{w}^T\phi(y_2, \mathbf{x})$ $\mathbf{w}^T\phi(y_3, \mathbf{x})$ $\mathbf{w}^T\phi(y_2, y_3)$

Each node is a random variable

We observe some nodes and need to assign the rest

Each factor is associated with a score

From scores to a probability

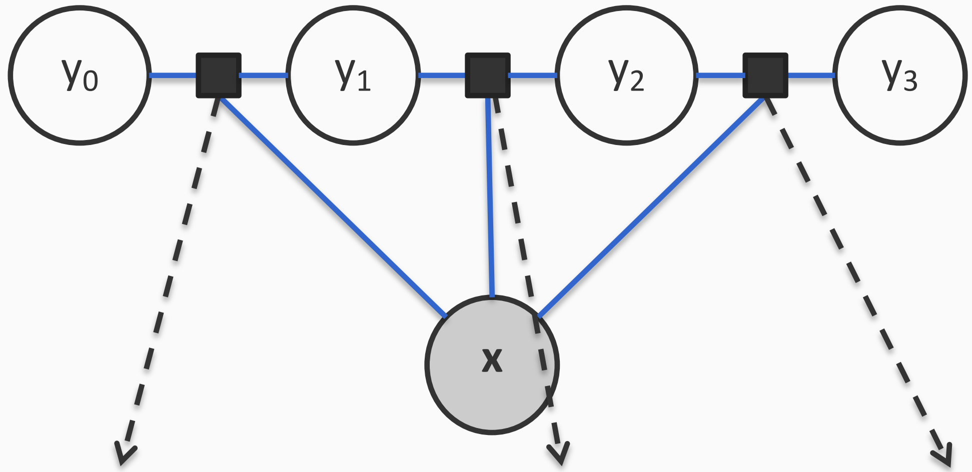


$\boldsymbol{w}^T\phi(\boldsymbol{x}, y_0, y_1)$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_1, y_2)$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_2, y_3)$

Recall our goal: To characterize a probability distribution over the unobserved variables, conditioned on the observed ones.

That is, to characterize $P(\, y_0, y_1, \cdots \mid \mathbf{x}\,)$.

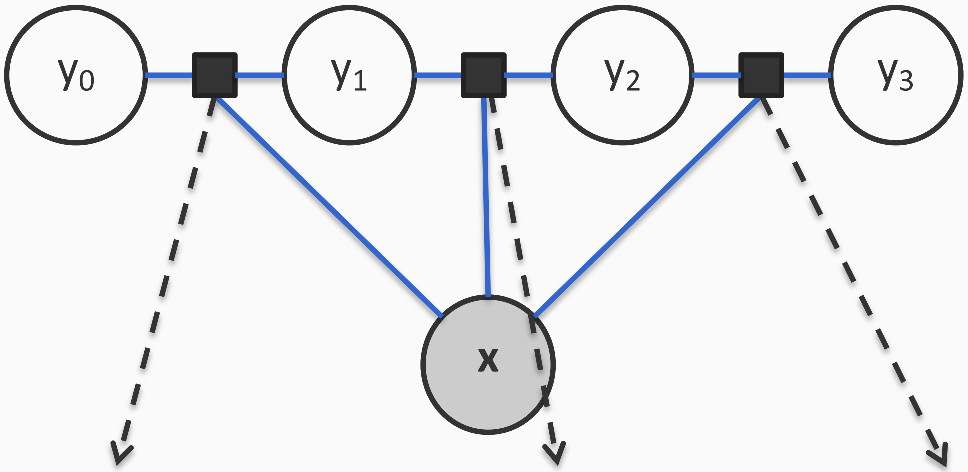
From scores to a probability



$\boldsymbol{w}^T\phi(\boldsymbol{x}, y_0, y_1)$ $\quad$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_1, y_2)$ $\quad$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_2, y_3)$

$$P(\mathbf{y} \mid \mathbf{x}) \propto \prod_{f \in \text{factors}} \exp\big(\text{score}(factor)\big)$$

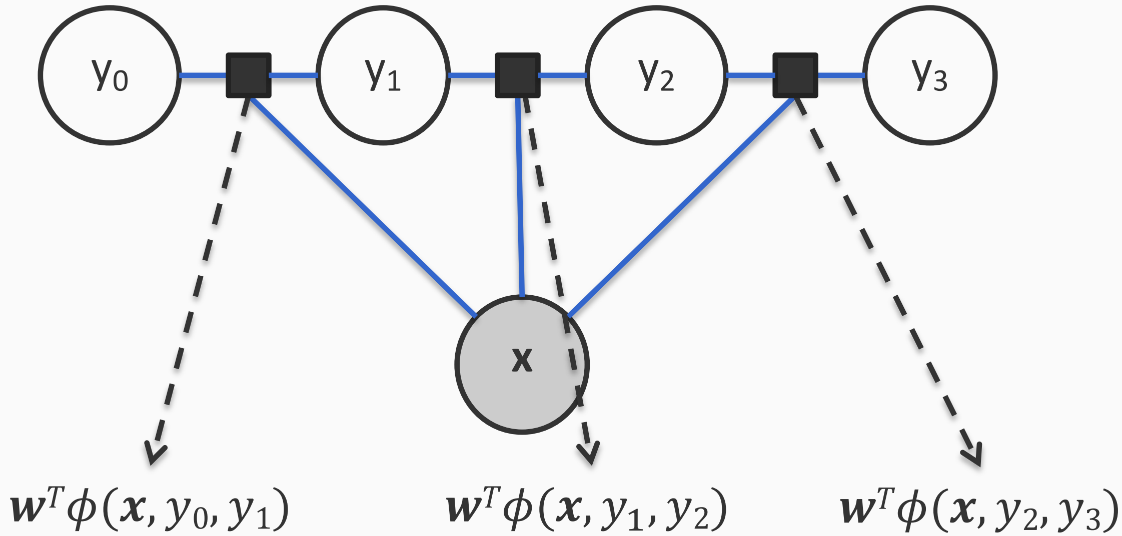
From scores to a probability



$\boldsymbol{w}^T\phi(\boldsymbol{x}, y_0, y_1)$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_1, y_2)$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_2, y_3)$

$$P(\mathbf{y} \mid \mathbf{x}) \propto \prod_i \exp\left(w^T \phi(\mathbf{x}, y_{i-1}, y_i)\right)$$

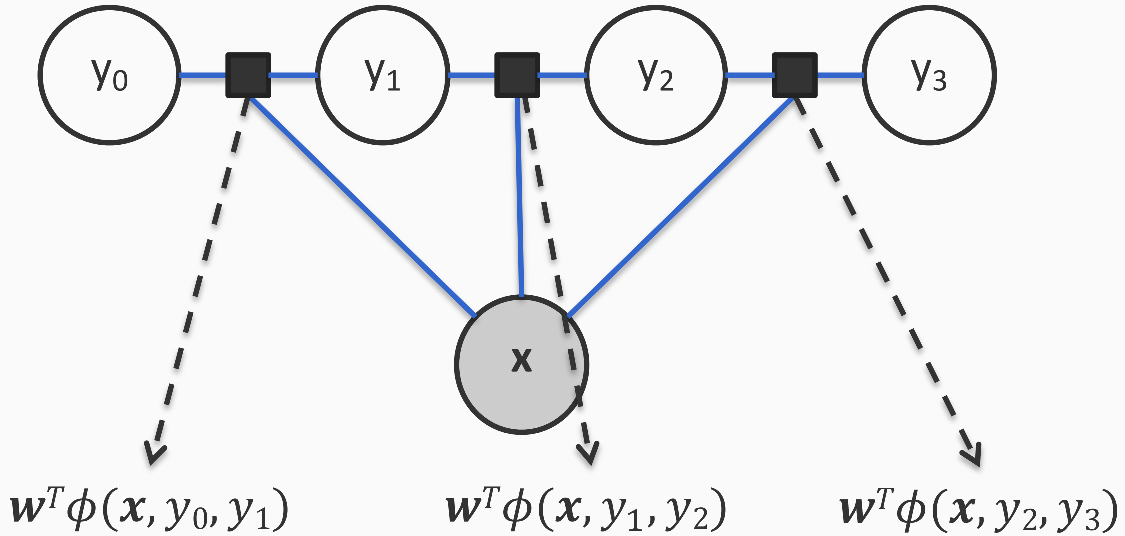
From scores to a probability



$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \exp\left(w^T \phi(\mathbf{x}, y_{i-1}, y_i)\right)$$

To get a probability, we need to normalize this using a term $Z(\mathbf{x})$ that ensures that the probabilities add up to one.

From scores to a probability



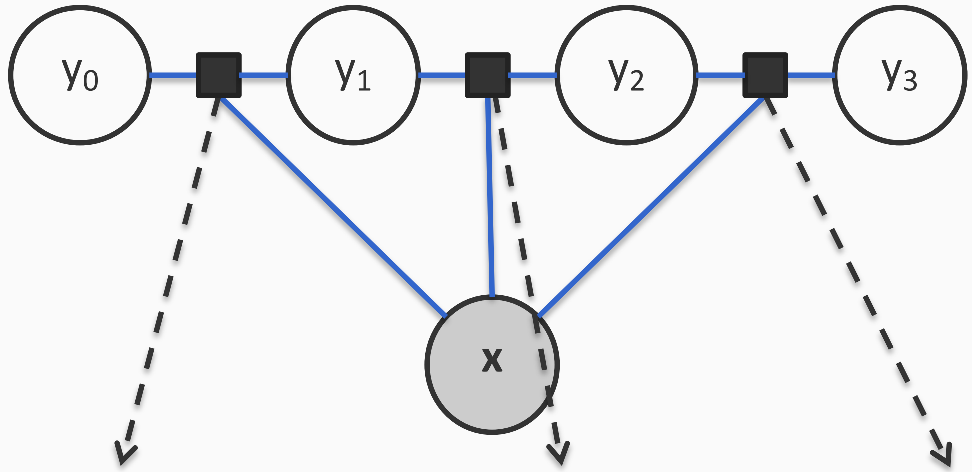
$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \exp\left(w^T \phi(\mathbf{x}, y_{i-1}, y_i)\right)$$

To get a probability, we need to normalize this using a term $Z(\mathbf{x})$ that ensures that the probabilities add up to one.

$$Z(\mathbf{x}) = \sum_{\hat{y}} \left(\prod_i \exp(w^T \phi(x, y_{i-1}, y_i) \right)$$

Called the *partition function*

Conditional Random Fields



$\boldsymbol{w}^T\phi(\boldsymbol{x}, y_0, y_1)$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_1, y_2)$ $\boldsymbol{w}^T\phi(\boldsymbol{x}, y_2, y_3)$

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \exp\left(w^T \phi(\mathbf{x}, y_{i-1}, y_i)\right)$$

The conditional probability of the labels given the input is a product of normalized factor scores.

Such models are called ***conditional random fields***.

CRF: A different view

- Input: $\mathbf{x}$, Output: $\mathbf{y}$, sequence (for now)
- Define a feature vector for the **entire** input and output sequence: $\Phi(\mathbf{x}, \mathbf{y})$

CRF: A different view

- Input: **x**, Output: **y**, sequence (for now)
- Define a feature vector for the **entire** input and output sequence: $\Phi(\mathbf{x}, \mathbf{y})$
- Define a giant log-linear model, P(**y** | **x**) parameterized by **w**

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} \prod_i \exp\left(\mathbf{w}^T \phi(\boldsymbol{x}, y_i, y_{i-1})\right) \propto \exp\left(\mathbf{w}^T \sum_i \phi(\boldsymbol{x}, y_i, y_{i-1})\right)$$

CRF: A different view

- Input: **x**, Output: **y**, sequence (for now)
- Define a feature vector for the **entire** input and output sequence: $\Phi(\mathbf{x}, \mathbf{y})$
- Define a giant log-linear model, P(**y** | **x**) parameterized by **w**

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z}\prod_i \exp\left(\mathbf{w}^T\phi(\boldsymbol{x}, y_i, y_{i-1})\right) \propto \exp\left(\mathbf{w}^T\sum_i \phi(\boldsymbol{x}, y_i, y_{i-1})\right)$$

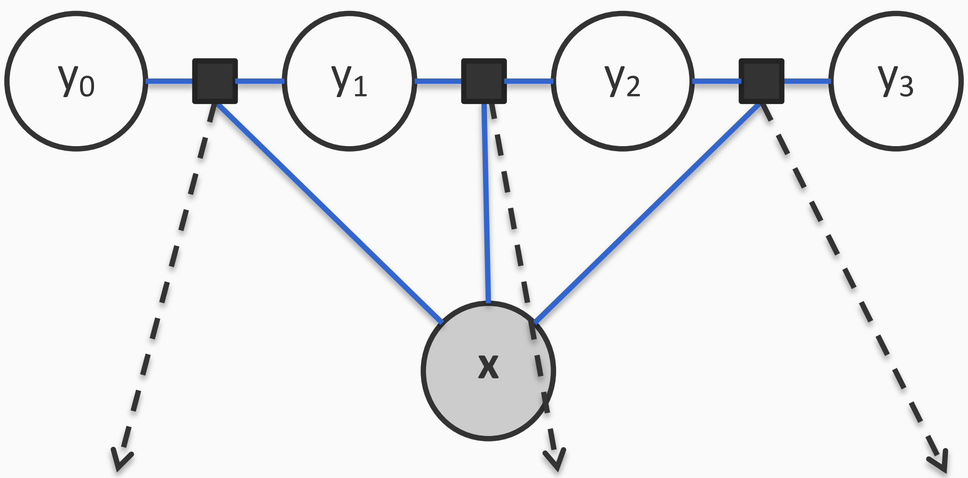
  – Just like any other log-linear model, except
    - Space of **y** is the set of all possible sequences of the correct length
    - Normalization constant sums over all sequences

In an MEMM, probabilities were locally normalized

Global features

The feature function decomposes over the factors in sequence (that is, the factor graph)

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_i \phi(x, y_{i-1}, y_i)$$



$\mathbf{w}^T\phi(\boldsymbol{x}, y_0, y_1)$ $\mathbf{w}^T\phi(\boldsymbol{x}, y_1, y_2)$ $\mathbf{w}^T\phi(\boldsymbol{x}, y_2, y_3)$

Where are we?

- We have seen how a CRF assigns probabilities to sequences
  - Global normalization instead of local normalization
  - Avoid the label bias problem because of this

- Next:
  - How to predict the most probable sequence
  - How to train the scoring functions

Prediction

Goal: To predict most probable sequence **y** for an input **x**

$$\operatorname*{argmax}_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \exp(\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}))$$

$$= \operatorname*{argmax}_{\mathbf{y}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})$$

Prediction

Goal: To predict most probable sequence **y** for an input **x**

$$\underset{\mathbf{y}}{\operatorname{argmax}}\, P(\,\mathbf{y} \mid \mathbf{x}\,) = \underset{\mathbf{y}}{\operatorname{argmax}} \exp(\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}))$$
$$= \underset{\mathrm{y}}{\operatorname{argmax}}\, \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})$$

But the score decomposes as $\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) = \sum_i \mathbf{w}^T \phi(\mathrm{x}, y_{i-1}, y_i)$

Prediction

Goal: To predict most probable sequence **y** for an input **x**

$$\underset{\mathbf{y}}{\operatorname{argmax}}\, P(\mathbf{y} \mid \mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}}\, \exp(\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}))$$
$$= \underset{\mathrm{y}}{\operatorname{argmax}}\, \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})$$

But the score decomposes as $\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) = \sum_i \mathbf{w}^T \phi(\mathrm{x}, y_{i-1}, y_i)$

Prediction via Viterbi (with sum instead of product)

Prediction

Goal: To predict most probable sequence **y** for an input **x**

$$\underset{\mathbf{y}}{\operatorname{argmax}} P(\mathbf{y} \mid \mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}} \exp(\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}))$$
$$= \underset{\mathrm{y}}{\operatorname{argmax}}\, \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})$$

But the score decomposes as $\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) = \sum_i \mathbf{w}^T \phi(\mathrm{x}, y_{i-1}, y_i)$

Prediction via Viterbi (with sum instead of product)

1. Base case: $\mathrm{score}_0(\mathrm{s}) = \mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}, \mathrm{start}, y_0)$

Prediction

Goal: To predict most probable sequence **y** for an input **x**

$$\operatorname*{argmax}_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \exp(\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}))$$
$$= \operatorname*{argmax}_{\mathrm{y}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})$$

But the score decomposes as $\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) = \sum_i \mathbf{w}^T \phi(\mathrm{x}, y_{i-1}, y_i)$

Prediction via Viterbi (with sum instead of product)

1. Base case: $\mathrm{score}_0(\mathrm{s}) = \mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}, \mathrm{start}, y_0)$
2. Recursive case:

$$\mathrm{score}_{\mathrm{i}}(\mathrm{s}) = \max_{y_{i-1}} (\mathbf{w}^T \phi(\mathbf{x}, y_{i-1}, y_i) + \mathrm{score}_{i-1}(y_{i-1}))$$

Training a chain CRF

- Input:
  - Dataset with labeled sequences, D = {<$\mathbf{x}_i$, $\mathbf{y}_i$>}
  - A definition of the feature function

- How do we train?
  - Maximize the (regularized) log-likelihood

$$\max_{\mathbf{w}} -\frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

Recall: Empirical loss minimization

Training with inference

$$\max_{\mathbf{w}} -\frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

- Many methods for training
  - Numerical optimization
  - Can use a gradient or hessian based method

- Simple gradient ascent

$$\mathbf{w} \leftarrow \mathbf{w} + \sum_i \left(\phi(\mathbf{x}_i, \mathbf{y}_i) - \sum_{\hat{\mathbf{y}}} P(\hat{\mathbf{y}}|\mathbf{x}_i, \mathbf{w}) \phi(\mathbf{x}_i, \hat{\mathbf{y}}) \right)$$

Training with inference

$$\max_{\mathbf{w}} -\frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

- Many methods for training
  - Numerical optimization
  - Can use a gradient or hessian based method

- Simple gradient ascent

$$\mathbf{w} \leftarrow \mathbf{w} + \sum_i \left(\phi(\mathbf{x}_i, \mathbf{y}_i) - \sum_{\hat{\mathbf{y}}} P(\hat{\mathbf{y}}|\mathbf{x}_i, \mathbf{w})\phi(\mathbf{x}_i, \hat{\mathbf{y}}) \right)$$

- Training involves inference!
  - A different kind than what we have seen so far
  - Summing over all sequences is just like Viterbi
    - With summation instead of maximization

CRF (for sequences): Summary

- An undirected graphical model
  - Decompose the score over the structure into a collection of factors
  - Each factor assigns a score to assignment of the random variables it is connected to

- Training and prediction
  - Final prediction via argmax $w^T\phi(\mathbf{x}, \mathbf{y})$
  - Train by maximum (regularized) likelihood

- Relation to other models
  - Effectively a linear classifier
  - A generalization of logistic regression to structures
  - An instance of Markov Random Field, with some random variables observed
    - We will see this soon