Recap: Markov network

- Markov network structure $\mathcal{H}$ for set of random variables
  $\rightarrow$ Encodes conditional independence assumptions
- Markov network $P$ over $\mathcal{H}$
  $\rightarrow$ Distribution in which these cond. indep. assumptions hold
- Factor graph
  $\rightarrow$ Factorization of $P$ in terms of local factors
- Log-linear model
  $\rightarrow$ Factorization of $P$ in terms of weighted feature functions

We now look at classification/regression

- Two types of random variables
  - $\mathcal{X}$: input variables (known at prediction time)
  - $\mathcal{Y}$: output variables (unknown, to be predicted)
  - Sometimes also $\mathcal{Z}$: nuisance variables (unknown, not to be predicted)
Markov networks for classification

- Markov network is distribution over $\mathcal{X} \cup \mathcal{Y}$
- E.g., Naive Bayes: $\mathcal{X} = \{X_1, \ldots, X_D\}$, $\mathcal{Y} = \{Y\}$

$\mathcal{H}_{\text{NBA}} =$

- To predict, we determine $P(Y | \mathcal{X})$
- Markov networks are generative models
  - Capture joint distribution $P(\mathcal{X}, \mathcal{Y})$
  - Thus: also capture prior class distribution $P(\mathcal{Y})$
  - Thus: also capture class-conditional densities $P(\mathcal{X} | \mathcal{Y})$
  - Thus: also capture input distribution $P(\mathcal{X})$

- Conditional independence assumptions refer to $\mathcal{X}$ and $\mathcal{Y}$
  - We model input $\mathcal{X}$ and output $\mathcal{Y}$
  - We make assumptions about $\mathcal{X}$ and $\mathcal{Y}$
Outline

1. Conditional Random Fields

2. Examples of CRFs

3. Linear-Chain CRFs

4. Training CRFs

5. Summary
Conditional random fields

- Input distribution $\mathcal{X}$ can be complex and difficult to model
  - Example: $\mathcal{X} = $ natural language sentence, $Y = $ spam?
  - Suppose each $X_j$ is a word
  - Probability of “You can earn a lot of money by simply answering this email”?
  - What comes after “The capital of Germany is”? 
  - Probability that third word is capitalized?

- To deal with complex inputs, we can
  1. Make assumptions about $\mathcal{X}$ to get a tractable model, or
  2. Use a discriminative model

- **Conditional random fields** (CRF) are discriminative graphical models
  - Model $P(\mathcal{Y} | \mathcal{X})$ only
  - Do not model input $\mathcal{X}$
  - Do not make assumptions about $\mathcal{X}$

- CRFs cannot answer the example questions above
  - But: we do not need to answer these questions to predict
Recap: Markov network graphs

- Consider the following Markov network graphs and their associated conditional independence statements:

\[\mathcal{H}_1\]

- \(X_1 \perp X_2 \mid Y\)
- \(X_1 \perp X_3 \mid Y\)
- \(X_2 \perp X_3 \mid Y\)
- \(X_1 \perp X_2, X_3 \mid Y\)
- \(X_1, X_2 \perp X_3 \mid Y\)
- \(X_1, X_3 \perp X_2 \mid Y\)
- \(X_2 \perp X_3 \mid Y, X_1\)
- \(X_1 \perp X_3 \mid Y, X_2\)
- \(X_1 \perp X_2 \mid Y, X_3\)

\[\mathcal{H}_2\]

- \(X_1 \perp X_2 \mid Y\)
- \(X_1 \perp X_3 \mid Y\)
- \(X_1 \perp X_2, X_3 \mid Y\)
- \(X_1 \perp X_3 \mid Y, X_2\)
- \(X_1 \perp X_2 \mid Y, X_3\)
Going discriminative

- Let’s strike out all assumptions that refer to the (conditional) distribution of $X = \{ X_1, X_2, X_3 \}$

\[ \mathcal{H}_1 \mid X \]

\[ Y \]

\[ X_1 \perp X_2 \mid Y \]
\[ X_1 \perp X_3 \mid Y \]
\[ X_2 \perp X_3 \mid Y \]
\[ X_1 \perp X_2, X_3 \mid Y \]
\[ X_1, X_2 \perp X_3 \mid Y \]
\[ X_1, X_3 \perp X_2 \mid Y \]
\[ X_2 \perp X_3 \mid Y, X_1 \]
\[ X_1 \perp X_3 \mid Y, X_2 \]
\[ X_1 \perp X_2 \mid Y, X_3 \]

\[ \mathcal{H}_2 \mid X \]

\[ Y \]

\[ X_1 \perp X_2 \mid Y \]
\[ X_1 \perp X_3 \mid Y \]
\[ X_1 \perp X_2, X_3 \mid Y \]
\[ X_1, X_3 \perp Y, X_2 \]
\[ X_1 \perp X_2 \mid Y, X_3 \]
**Conditional Markov network graph**

**Definition**

Let $\mathcal{H}$ be a Markov network graph over variables $\mathcal{X} \cup \mathcal{Y}$. We associate with the **conditional Markov network graph** $\mathcal{H} | \mathcal{X}$ the subset of conditional independence statements from $\mathcal{H}$ that are associated with the Markov network graph $\mathcal{H}'$ obtained from $\mathcal{H}$ by connecting all pairs of variables in $\mathcal{X}$.

- All cond. indep. statements that include all variables in $\mathcal{X}$ in their condition are retained
  - These are cond. indep. stmts. between variables in $\mathcal{Y}$
  - None if there is only one variable in $\mathcal{Y}$
  - Most useful when there is more than one variable in $\mathcal{Y}$

- No cond. indep. stmts. between variables in $\mathcal{X}$ retained
  - Edges between variables in $\mathcal{X}$ in $\mathcal{H}$ don’t matter
  - So we don’t draw them (like in $\mathcal{H}_1 | \mathcal{X}$) and mark the variables in $\mathcal{X}'$ with gray background
Example

- Note: $\mathcal{H}_1 | \mathcal{X}$ models, for example, logistic regression

- Another example

- Which statements remain? → tutorial
Revisiting factor graphs

- Recall: factor graphs visualize factorization of a distribution

$$X_1 \times X_2 \times X_3$$

- We can read off conditional independence statements by “converting” factor graphs to Markov network graphs
  - For each factor, connect all vertices in its scope
  - Gives smallest Markov network graph for which set of factors is consistent

- Factor graphs provide more detail
  - Markov network graphs describes which factorizations are allowed
  - Factor graph describes which allowed factors are actually used
Conditional random field (CRF)

**Definition**

Let $G = (\mathcal{X} \cup \mathcal{Y}, \mathcal{F})$ be a factor graph over variables $\mathcal{X} \cup \mathcal{Y}$ and with factors $\mathcal{F}$. Then $P(\mathcal{Y} | \mathcal{X})$ is a **conditional random field (CRF)** w.r.t. $G$ if for any value $x$ of $\mathcal{X}$, distribution $P(\mathcal{Y} | x)$ factorizes according to $G$:

$$P(y | x) = \frac{1}{Z(x)} \prod_{\phi \in \mathcal{F}} \phi(x_\phi, y_\phi).$$

- CRFs only need to model $P(\mathcal{Y} | \mathcal{X})$.
- Normalization constant now depends on $x$.
  - $Z(x) = \sum_{y'} \prod_{\phi \in \mathcal{F}} \phi(x_\phi, y'_\phi)$
  - Sometimes easier to compute than for Markov networks.
- As before, we will indicate CRFs by marking variables in $\mathcal{X}$ with gray background.
Factorizations and classification (1)

- Consider any factor graph $\mathcal{G} = (\mathcal{X} \cup \mathcal{Y}, \mathcal{F})$
- Let $\mathcal{F}_\mathcal{X} \subseteq \mathcal{F}$ be the set of of factors that only refer to variables in $\mathcal{X}$
- Let’s look at prediction:

$$P(y \mid x) = \frac{P(y, x)}{P(x)} = \cdots = \frac{\prod_{\phi \in \mathcal{F}} \phi(x_\phi, y_\phi)}{\sum_{y'} \prod_{\phi \in \mathcal{F}} \phi(x_\phi, y'_\phi)} = \frac{\prod_{\phi \in \mathcal{F} \setminus \mathcal{F}_\mathcal{X}} \phi(x_\phi, y_\phi) \prod_{\phi \in \mathcal{F}_\mathcal{X}} \phi(x_\phi)}{\sum_{y'} \prod_{\phi \in \mathcal{F} \setminus \mathcal{F}_\mathcal{X}} \phi(x_\phi, y'_\phi) \prod_{\phi \in \mathcal{F}_\mathcal{X}} \phi(x_\phi)}$$

(separate out $\mathcal{F}_\mathcal{X}$)

- Conclusion: factors in $\mathcal{F}_\mathcal{X}$ do not matter for prediction
Factorizations and classification (2)

- In left factor graph:
  - $\mathcal{F} = \{ \phi_1, \phi_2, \phi_3, \psi_1, \psi_2 \}$
  - $\mathcal{F}_X = \{ \phi_1, \phi_2, \phi_3 \}$

- Both factor graphs agree on $P(\mathcal{Y} | \mathcal{X})$
  - But they disagree on, say, $P(\mathcal{X})$

- In CRFs, we only model $P(\mathcal{Y} | \mathcal{X})$
  - All factors in $\mathcal{X}_\mathcal{F}$ can be ignored
  - In fact, we would not even bother to create them in the first place
Parameterized factors

- Factors are often parameterized, i.e., the function they compute depends on a subset of some parameter set $\theta$

**Categorical Naive Bayes**

- $Y$ is the output
- $\pi$ represents the class distribution
- $\theta_1, \theta_2, \theta_D$ are parameters
- $X_1, X_2, \ldots, X_D$ are inputs

**Logistic regression**

- $Y$ is the output
- $w_1, w_2, w_D$ are parameters
- $X_1, X_2, \ldots, X_D$ are inputs

- For logistic regression: $\phi(x_j, 1; w_j) = \exp(w_jx_j)$ and $\phi(x_j, 0; w_j) = 1$

- To fit a model, we can (in principle) proceed as before; e.g.,
  - Maximum likelihood estimation for $\theta$
  - MAP estimation for $\theta$
  - Bayesian inference (often done by creating variables for $\theta$)
Outline

1. Conditional Random Fields

2. Examples of CRFs

3. Linear-Chain CRFs

4. Training CRFs

5. Summary
Sequence labeling (1)

- CRFs are often used when they are multiple output variables
- The prototypical CRF application is **sequence labeling**
  - Input variables form a sequence
  - One output variable per input → also form a sequence
- Examples
  - Part-of-speech tagging
    
    | $i$ | 1 | 2 | 3 | 4 | 5 |
    |-----|---|---|---|---|---|
    | $Y_i$ | DT | NN | VBD | DT | NN |
    | $X_i$ | The dog ate the cat |
  - Other NLP tasks (e.g., noun-phrase chunking, named entity recognition, ...)
  - Bioinformatics (e.g., protein structure prediction)
  - Handwriting recognition
  - Speech recognition
  - ...
Sequence labeling (2)

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_i$</td>
<td>DT</td>
<td>NN</td>
<td>VBD</td>
<td>DT</td>
<td>NN</td>
</tr>
<tr>
<td>$X_i$</td>
<td>The dog ate the cat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- How to model?
- E.g., dependency structure
  1. Assume labels cond. independent given inputs
     → Easy to learn, often not a suitable choice (see above example)
  2. Assume full dependence
     → Difficult to learn, can capture arbitrary dependencies
  3. Something in between
- E.g., type of model
  1. Generative
     → Need to (or is able to) make assumptions about inputs
  2. Discriminative
     → No assumptions about input
Example: simple dependency structure, generative

This model assumes that
- Input $X_i$ influenced directly only by associated output $Y_i$
  \[(X_i \perp (X \cup Y) \setminus \{X_i, Y_i\} | Y_i)\]
- Outputs directly influenced only by neighboring outputs and corresponding input
  \[(Y_i \perp (X \cup Y) \setminus \{X_i, Y_{i-1}, Y_i, Y_{i+1}\} | Y_{i-1}, X_i, Y_{i+1})\]
- These assumptions are quite strong (esp. the first one)

Example of a **Hidden Markov Model**
- $Y$ called **hidden states**
- $X$ called **observed outputs** (note the difference in terminology)
Example: simple dependency structure, discriminative

\[ \mathcal{Y} \xrightarrow{DT} \mathcal{X} \xrightarrow{NN} \mathcal{Y} \xrightarrow{VBD} \mathcal{X} \xrightarrow{DT} \mathcal{Y} \]

- This model assumes that
  - Input \( X_i \) influenced directly only by associated output \( Y_i \)
    \[ (X_i \perp (\mathcal{X} \cup \mathcal{Y}) \setminus \{X_i, Y_i\} \mid Y_i) \]
  - Outputs directly influenced only by neighboring outputs and corresponding input
    \[ (Y_i \perp (\mathcal{X} \cup \mathcal{Y}) \setminus \{X_i, Y_{i-1}, Y_i, Y_{i+1}\} \mid Y_{i-1}, X_i, Y_{i+1}) \]

- Example of a linear-chain CRF
  - Conditioned on \( \mathcal{X} \)
  - Output variables form a single “chain”
Example: more complex dependencies, discriminative

- This model assumes that
  - Input $X_i$ influenced directly only by associated output $Y_i$
    \[(X_i \perp (X \cup Y) \setminus \{ X_i, Y_i \} \mid Y_i)\]
  - Outputs directly influenced only by neighboring outputs and all inputs
    \[(Y_i \perp Y \setminus \{ Y_{i-1}, Y_i, Y_{i+1} \} \mid Y_{i-1}, X, Y_{i+1})\]
- General form of linear-chain CRF
Example: Long-Range Dependencies

- Here we added **long-range dependencies**
  - Can now model that same words should (be more likely to) share label
- That’s not a linear-chain CRF anymore
Example: Two-Chains CRF

- Here $\mathcal{Y}_2$ performs noun-phrase chunking
  - B-NP/I-NP mean beginning of/in noun phrase
  - O means out of noun phrase
- This two-chains CRF performs **multi-task learning**
  - Solve multiple tasks jointly
  - Exploit mutual reinforcement
  - Here: knowledge of POS tags helps noun-phrase chunking
  - Here: knowledge of noun-phrase chunks helps POS tagging
Because a generative model takes the form $p(y, x) = p(y)p(x|y)$, it is often natural to represent a generative model by a directed graph in which the outputs $y$ topologically precede the inputs. Similarly, we will see that it is often natural to represent a discriminative model by an undirected graph. However, this need not always be the case, and both undirected generative models, such as the Markov random field (2.32), and directed discriminative models, such as the MEMM (6.2), are sometimes used. It can also be useful to depict discriminative models by directed graphs in which the $x$ precede the $y$.

The relationship between naive Bayes and logistic regression mirrors the relationship between HMMs and linear-chain CRFs. Just as naive Bayes and logistic regression are a generative-discriminative pair, there is a discriminative analogue to the HMM, and this analogue is a particular special case of CRF, as we explain in the next section. This analogy between naive Bayes, logistic regression, generative models, and CRFs is depicted in Figure 2.4.

Fig. 2.4 Diagram of the relationship between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs.
Conditional Markov Logic Networks (1)

- **Conditional Markov logic networks** are CRFs that condition on “input relations”
- Example of a general CRF
- Features expressed in terms of logical formulas
- Application: analyse citations in scientific publications
  - Such as this one


- Input: words and symbols in citation (*Token* relation)
- Output: field of a token
  (*FieldIn* relation; e.g., author, publication name, venue, ...)
- Output: whether two citations agree on a field
  (*SameField* relation)
- Output: whether two citations refer to the same publication
  (*SameCit* relation)
Conditional Markov Logic Networks (2)

Token\((token, position, citation)\)
InField\((position, field, citation)\)
SameField\((field, citation, citation)\)
SameCit\((citation, citation)\)

Token\((+t, i, c)\) => InField\((i, +f, c)\)
InField\((i, +f, c)\) <=> InField\((i + 1, +f, c)\)
\(f_1 \neq f_2 \Rightarrow (\neg InField(i, +f_1, c) \lor \neg InField(i, +f_2, c))\)

Token\((+t, i_1, c_1)\) \land InField\((i_1, +f, c_1)\)
\land Token\((+t, i_2, c_2)\) \land InField\((i_2, +f, c_2)\)
=> SameField\((+f, c_1, c_2)\)
SameField\((+f, c_1, c_2)\) <=> SameCit\((c_1, c_2)\)
SameField\((f, c_1, c_2)\) \land SameField\((f, c_2, c_3)\)
=> SameField\((f, c_1, c_3)\)
SameCit\((c_1, c_2)\) \land SameCit\((c_2, c_3)\) => SameCit\((c_1, c_3)\)
Outline

1. Conditional Random Fields

2. Examples of CRFs

3. Linear-Chain CRFs

4. Training CRFs

5. Summary
Factor graph

- Recall: a general linear-chain CRF has the following form:

\begin{align*}
  y & \rightarrow DT \rightarrow NN \rightarrow VBD \rightarrow DT \rightarrow NN \\
  x & \rightarrow The \rightarrow dog \rightarrow ate \rightarrow the \rightarrow cat
\end{align*}

- We can alternatively write this as a factor graph:

\begin{align*}
  y & \rightarrow DT \rightarrow NN \rightarrow VBD \rightarrow DT \rightarrow NN \\
  x & \rightarrow The \rightarrow dog \rightarrow ate \rightarrow the \rightarrow cat
\end{align*}
Log-linear model

When using a linear-chain CRF, we usually

- Use features instead of factors
- I.e., we use a (conditional) log-linear model to specify the CRF

\[
P(\mathcal{Y} | \mathcal{X}) \propto \prod_j \exp[w_j f_j(\mathcal{X}_j, \mathcal{Y}_j)] = \exp\left[\sum_j w_j f_j(\mathcal{X}_j, \mathcal{Y}_j)\right]
\]

- Each feature either involves either one output or two neighboring outputs, and a (possibly empty) subset of inputs \( \mathcal{X} \)

- Training = fit weights
- Modeling = provide useful features (feature engineering)
  - Since model is discriminative, we can use arbitrary features
- Weights tell us how “useful” a feature is
  - E.g.: indicator feature saying that \( i \)-th input is “the” and \( i \)-th output is “DET”
  - Positive weight \( \rightarrow \) “the” encourages output “DET”
  - Negative weight \( \rightarrow \) “the” discourages output “DET”
Template models

- In many sequence labeling tasks
  1. Sequences can have different lengths
  2. Features are local and (often) do not depend on position

- When we use CRFs for sequence labeling
  - We use feature templates to construct a CRF for an input \( x \)
  - We use parameter tying to share weights across features

- Example: feature template for unigram features
  - For each \( i \), each input value \( x \), and each output value \( y \):
    - Add a factor \( \phi_{ixy}(X_i, Y_i \mid w_{xy}) = \exp(w_{xy} I_{X_i=x \land Y_i=y}) \)
    - \( I_{X_i=x, Y_i=y} \) is a feature and \( w_{xy} \) its weight
      - Factor \( \phi_{ixy} \) is parameterized by weight \( w_{xy} \)
    - The weight is shared across positions, but not for different \( x \) and \( y \)

- There often are many features
  - But: we do not need to instantiate a feature that it is always zero
    - Has no impact on \( P(y \mid x) \)
  - E.g., instantiate only if \( X_i = x \) and \( Y_i = y \) during training
  - E.g., instantiate only if \( X_i = x \) during prediction
Example (training)

- Here weight parameter is shown within the factor vertices
- Used indicator features
  - Unigram features (e.g., $w_{\text{the,DT}}$)
  - Features for each combination of neighboring outputs (e.g., $w_{\text{DT,NN}}$)
  - Features for start/end of sequence (e.g., $s_{\text{DT}}$)
  - Bias features for each class (e.g., $w_{\text{DT}}$)
Discussion

• Different inputs $\mathbf{x}$ lead to different factor graphs
  ▶ But all are parameterized by the same set of weights
  ▶ All weights together form the parameter vector $\mathbf{\theta}$
• Various features are often combined for efficiency reasons
  ▶ E.g., for unigram feature, $\mathbf{\theta}$ contains all input-output weights $w_{xy}$
  ▶ Combined template: For all $i$, add factor

$$
\phi_{\text{uni}}(X_i, Y_i \mid \mathbf{\theta}) = \exp \left[ \sum_{x,y} w_{xy} I_{X_i=x \land Y_i=y} \right] = \exp[w_{X_iY_i}]
$$

• For our example model, we ultimately obtain
Outline

1. Conditional Random Fields

2. Examples of CRFs

3. Linear-Chain CRFs

4. Training CRFs

5. Summary
Inference and training (1)

- Consider a general linear-chain CRF

\[
P(y | x, w) \propto \exp \left[ \sum_j w_j f_j(x, y_{kj}, y_{kj+1}) \right]
\]

- \(f_j\) is a feature
- Each feature uses all input variables
- \(y_{kj}\) and \(y_{kj+1}\) refer to the output variables being used
- \(w_j\) is weight of feature \(f_j\)
- \(w = (w_1 \ w_2 \ \ldots \ w_D)^T\) is the weight vector (= parameters of the model)
Inference and training (2)

\[ P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) \propto \exp \left[ \sum_j w_j f_j(\mathbf{x}, y_{k_j}, y_{k_j+1}) \right] \]

- **Inference**
  - Given an example \( \mathbf{x} \)
  - Determine \( P(\mathcal{Y} \mid \mathbf{x}, \hat{\mathbf{w}}) \) (for point estimate \( \hat{\mathbf{w}} \) of \( \mathbf{w} \))
  - Determine \( P(\mathcal{Y} \mid \mathbf{x}) \) (for Bayesian inference)
  - Determine \( \text{argmax}_y P(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{w}}) \) (for point estimate \( \hat{\mathbf{w}} \) of \( \mathbf{w} \))
  - Determine \( \text{argmax}_y P(\mathbf{y} \mid \mathbf{x}) \) (for Bayesian inference)
  - ...

- **Training**
  - Given a dataset \( \mathcal{D} = \{ (\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N) \} \)
  - Determine \( \hat{\mathbf{w}}_{\text{MLE}} \) (for maximum likelihood)
  - Determine \( \hat{\mathbf{w}}_{\text{MAP}} \) (for maximum a posteriori)
  - Determine \( P(\mathbf{w} \mid \mathcal{D}) \) (for Bayesian inference)

- **Here**
  - We discuss maximum likelihood estimation
  - We assume that each input has the same length
Maximum likelihood estimation

- We want to maximize the conditional likelihood

\[
P(Y \mid X, w) = \prod_i \frac{1}{Z(x_i \mid w)} \exp \left[ \sum_j w_j f_j(x_i, y_{ikj}, y_{i(kj+1)}) \right]
\]

- Here: \(Z(x_i \mid w) = \sum_{y' \in \text{Val}(Y)} \exp \left[ \sum_j w_j f_j(x_i, y_{ikj}, y_{i(kj+1)}) \right]\)

- As before, we can maximize the cond. log-likelihood instead

\[
\ell(Y \mid X, w) = \sum_i \left[ \sum_j w_j f_j(x_i, y_{ikj}, y_{i(kj+1)}) - \ln Z(x_i \mid w) \right]
\]

- This gives us the following optimization problem

\[\hat{w}_{\text{MLE}} = \arg\max_w \ell(Y \mid X, w)\]

  - Note that \(\ln Z(x_i \mid w)\) depends on \(w\) and thus cannot be ignored
Taking gradients

\[
\begin{align*}
\frac{\partial}{\partial w_j} \ell(Y \mid X, w) &= \frac{\partial}{\partial w_j} \left[ \sum_i \sum_j w_j f_j(x_i, y_{ikj}, y_i(k_j+1)) \right. \\
&\quad \left. - \sum_i \ln Z(x_i \mid w) \right] \\
&= \frac{\partial}{\partial w_j} \left[ \sum_i \sum_j w_j f_j(x_i, y_{ikj}, y_i(k_j+1)) \right] - \sum_i \ln Z(x_i \mid w) \\
\frac{\partial}{\partial w_j} \tilde{T} &= \sum_i f_j(x_i, y_{ikj}, y_i(k_j+1)) \\
\frac{\partial}{\partial w_j} T &= \sum_i \sum_{y_i' \in \text{Val}(Y')} P(y_i' \mid x_i, w) f_j(x, y_{ikj}, y_i'(k_j+1))
\end{align*}
\]
Interpreting gradients (1)

- The expected value of $f_j$ under the **empirical distribution** is

$$\tilde{E}[f_j] = \frac{1}{N} \sum_i f_j(x_i, y_{ik}, y_{i(k+1)})$$

  - Empirical distribution = joint distribution of inputs and outputs in training data
  - That’s the mean value of $f_j$ in the training data
  - That’s proportional to the gradient of $\tilde{T}$

- The expected value of $f_j$ under the **model distribution** is

$$E[f_j] = \frac{1}{N} \sum_i \sum_{y_i'\in\text{Val}(Y)} P(y_i' \mid x_i, w) f_j(x_i, y_{ik}, y_{i(k+1)})$$

  - Model distribution (for discriminative model) = joint distribution of inputs from training data and outputs from model
  - That’s the expected value of $f_j$ for training examples according to the model
  - That’s proportional to the gradient of $T$
Interpreting gradients (2)

- We obtain
  \[
  \frac{\partial}{\partial w_j} \ell(Y | X, w) \propto \tilde{E}[f_j] - E[f_j]
  \]

- When we increase weight \( w_j \)
  - Feature \( f_j \) becomes more “important”
  - Values of \( Y \) for which \( f_j \) is large become more likely

- Suppose we use gradient ascent (GA)
  - GA increases \( w_j \) when the gradient is positive
    \( \rightarrow \) model underestimates “importance” of \( f_j \) in that \( E[f_j] < \tilde{E}[f_j] \)
  - GA decreases \( w_j \) when the gradient is negative
    \( \rightarrow \) model overestimates “importance” of \( f_j \) in that \( E[f_j] > \tilde{E}[f_j] \)

- This interpretation is not specific to linear-chain CRFs: it holds for all log-linear models
Example: Logistic regression

- Logistic regression is a CRF, too
  - It’s even a special case of linear-chain CRFs
- The feature functions are $f_j(x_j, y) = yx_j$ (see slide 14)
- Starting from the gradient we had derived:

$$
\frac{\partial}{\partial w_j} \ell(y \mid X, w) = \sum_{i=1}^{N} x_{ij} [y_i - p_{i1}]
$$

$$
= \sum_{i=1}^{N} y_i x_{ij} - \sum_{i=1}^{N} [p_{i0}0x_{ij} + p_{i1}1x_{ij}]
$$

$$
= \sum_{i=1}^{N} f_j(x_{ij}, y_i) - \sum_{i=1}^{N} \sum_{y_i' \in \{0, 1\}} p_{iy_i'} f_j(x_{ij}, y_i')
$$

$$
\propto \tilde{E}[f_j] - E[f_j]
$$
Efficient gradient computation (1)

- We can precompute the feature statistics $\tilde{E}[f_j]$ once.

- But how to compute the expectation under the model distribution?

$$E[f_j] = \frac{1}{N} \sum_i \sum_{y'_i \in \text{Val}(Y)} P(y'_i | x_i, w) f_j(x_i, y'_{ik_j}, y'_{i(k_j+1)})$$

  - Feasible is $|\text{Val}(Y)|$ is small (e.g., $=2$ for logistic regression).
  - Generally, $|\text{Val}(Y)|$ is exponentially large in the number of outputs.

- Similar arguments hold for inference:

$$P(y | x, w) = \frac{1}{Z(x | w)} \exp \left[ \sum_j w_j f_j(x, y_{k_j}, y_{k_j+1}) \right],$$

where $Z(x | w) = \sum_{y' \in \text{Val}(Y)} \exp[\sum_j w_j f_j(x, y'_{k_j}, y'_{k_j+1})]$.
Efficient gradient computation (2)

- For linear-chain CRFs
  - Restrictions on feature functions help
  - Training can be done efficiently using the **sum-product algorithm**
  - MAP prediction can be done efficiently using the **max-product algorithm**

- In general
  - Inference as discussed for Markov networks (e.g., loopy belief propagation, Gibbs samplings, ...)
  - Once we can do inference, we can (for example) sample from the model distribution
    → I.e., we can estimate $E[f_j]$
Outline

1. Conditional Random Fields
2. Examples of CRFs
3. Linear-Chain CRFs
4. Training CRFs
5. Summary
Lessons learned

- **Conditional random fields**
  - Wide class of powerful discriminative graphical models
  - Model $P(\mathcal{Y} | \mathcal{X})$ using a graphical model
  - Do not model $P(\mathcal{X})$

- **Modeling**
  - Log-linear model: features + weights
  - Templates to generate CRF for a particular input
  - Parameter tying to share feature weights across different instances of the “same” feature

- **Linear-chain CRFs**
  - Common model for sequence labeling
  - Efficient training and inference methods exist

- **Training and inference**
  - For all log-linear models, gradient of conditional log-likelihood = feature mean - feature expectation
  - Feature expectation often intractable to compute exactly
    - approximate inference methods
  - Approximate inference methods can directly be used for training
Suggested reading

- C. Sutton and A. McCallum
  *An introduction to conditional random fields for relational learning*
  Introduction to Statistical Relational Learning, pp. 93–128, 2006

- R. Klinger and K. Tomanek
  *Classical probabilistic models and conditional random fields*
  Algorithm Engineering, 2007

- C. Elkan
  *Log-linear models and conditional random fields*
  Tutorial notes at CIKM 2008 (also video lectures)

- K.P. Murphy, Y. Weiss, and M.I. Jordan
  *Loopy belief propagation for approximate inference: An empirical study*
  UAI 1999