Data analysis and stochastic modeling

Lecture 7 – Entropy and Conditional Random Fields

Guillaume Gravier

guillaume.gravier@irisa.fr

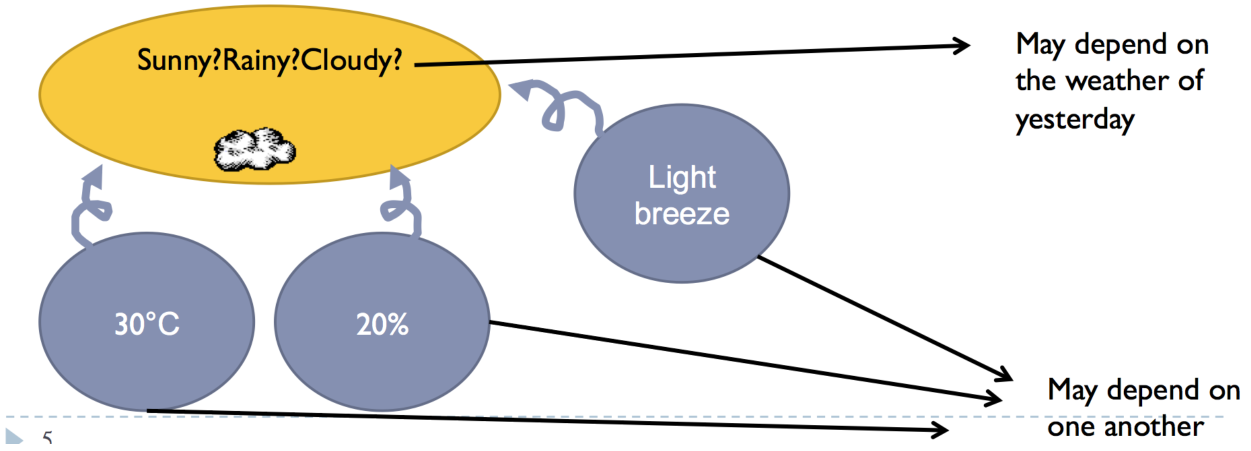
Maximum a posteriori classification

Given observed data x, we wish to predict a (discrete) label y. Bayesian (optimal) decision says that

$$\hat{y} = \arg\max_y p(y|x) = \arg\max_y p(x,y) = \arg\max_y p(x|y)p(y)$$

Example:

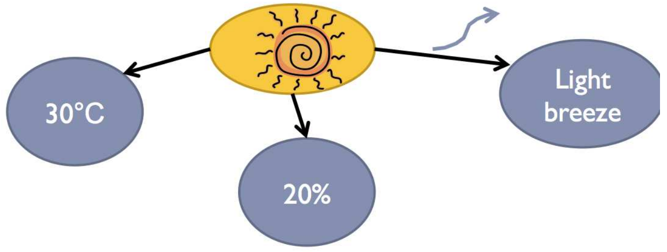
- y = weather of the day
- x = temprature, humidity, etc.


Sunny?Rainy?Cloudy?
May depend on the weather of yesterday
Light breeze
30°C
20%
May depend on one another
5

Maximum a posteriori classification (cont'd)

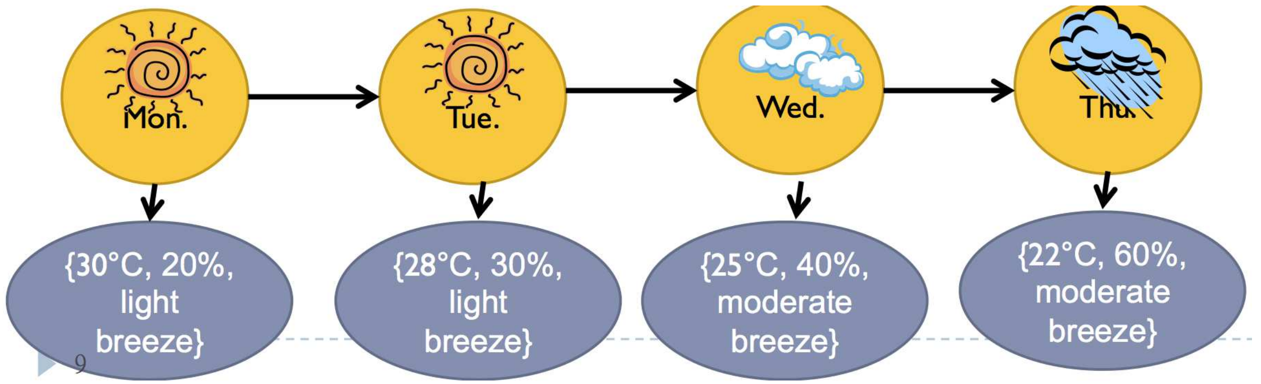
Classification from multiple attributes often makes independence assumption

$$p(\mathbf{x}, y) = p(y) \prod_{i=1}^{n} p(x_i | y)$$



Same holds for HMMs

$$p(\mathbf{x}, \mathbf{y}) = p(y_0) p(x_0 | y_0) \prod_{i=2}^{n} p(y_i | y_{i-1}) p(x_i | y_i)$$


Mon.
Tue.
Wed.
Thu.
{30°C, 20%, light breeze}
{28°C, 30%, light breeze}
{25°C, 40%, moderate breeze}
{22°C, 60%, moderate breeze}
9

The “discriminative” principle

Directly model the distribution of $p(y|\mathbf{x})$ to

- not waste effort on modeling the distribution of $\mathbf{x}$
- take into account interacting features and long term dependencies
- training directly for the task at hand

$\Rightarrow$ discriminative modeling is linked with maximum entropy distributions!

Usually opposing “generative” and “discriminative” models.

Entropy: Measuring uncertainty

The probability distribution of an event and the amount of information each event brings forms a random variable whose expected value is the average amount of information, called entropy, that the distribution is providing.

- deterministic event $\rightarrow$ entropy is null
- uniform distribution $\rightarrow$ entropy is one

Formally

$$H(X) = E[-\log_b P(x)] = -\sum_i P(x_i)\log_b P(x_i)$$

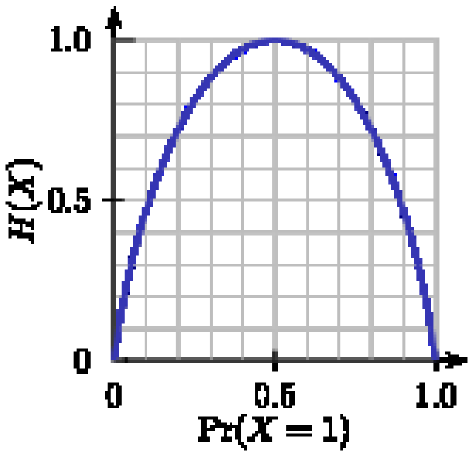
The Bernoulli process example: tossing a coin, varying $P[X = \text{heads}]$ from 0 to 1.

Conditional entropy:

$$H(X|Y) = \sum_{i,j} P(x_i, y_j)\log_b \frac{p(y_j)}{p(x_i, y_j)}$$

Claude E. Shannon

1916–2001



The maximum entropy principle

Definition: Given some testable information (or prior) we know about the data, the best probability distribution for the data is the one whose entropy is maximal subject to the constraints that we have from the priors.

Testable information can be

- knowledge of some moments, e.g., the mean of x is 0.25
- prior knowledge on probabilities, e.g., p(1) = 0.25 or p(1) + p(3) = 0.4

$\rightarrow$ often given by empirical estimates on training data for model specification

Examples of maximum entropy distributions you know:

- Uniform distribution and piecewise uniform distributions
- Normal distribution for a given mean and standard deviation
- Exponential distribution with given mean $1/\lambda$

Maximum entropy distribution illustrated

Example taken from A. Berger, S. Della Pietra and V. Della Pietra, A maximum entropy approach to natural language processing, Journal in Computational Linguistics, 22(1):39–71, 1996.]

Suppose we want to translate the English word *in* to the French language. To help us, we gather data in which the word *in* appears along with its translation. We observe that possible translation are {*dans, en, à, au cours de, pendant*}. We consider a simple probabilistic translation model to assign the probability for a translation of *in*, possibly given the context in which the word occurs.

Case 1: Nothing is known apart from the above and we only have the constraint that

$$p(\text{dans}) + p(\text{en}) + p(\text{à}) + p(\text{au cours de}) + p(\text{pendant}) = 1 \ .$$

The best model (the one with highest entropy) is uniform distribution.

Maximum entropy distribution illustrated (cont'd)

Case 2: We now observe empirically from the data that

$$p(\text{dans}) + p(\text{en}) = 3/10 \ .$$

The best model (the one with highest entropy) reflecting this new fact (plus the sum to one constraint) is piecewise uniform, i.e.,

$$p(\text{dans}) = p(\text{en}) = 3/20 \ \text{ and } \ p(\text{à}) = p(\text{au cours de}) = p(\text{pendant}) = 7/20$$

Case 3: We furthermore observe that

$$p(\text{dans}) + p(\text{à}) = 1/2 \ .$$

It's not so trivial anymore to find the best distribution!

$\Rightarrow$ We want to generalize the idea to select the best models consistent with facts that are observed on the training data, where facts can also be contextual, e.g., "if April is following the word *in*, we observe an emprirical frequency $\tilde{p}(\text{en}) = 0.9$".

Maximum entropy distribution: Back to theory

Consider a variable x taking values in $\{x_1, \ldots, x_n\}$ (not to be confused with training data). Constraints are expressed as m measurable feature functions $f_k(x)$ whose theoretical expectation should match the facts (e.g., the empirical frequencies in the previous example), i.e.,

$$\sum_{i=1}^{n} f_k(x_i) P(x_i) = F_k \quad \forall k = 1, \ldots, m \ .$$

The maximum entropy distribution for X has the form

$$P[X = x_k] = \frac{1}{Z} \exp\left(\sum_{j=1}^{m} \lambda_j f_j(x_k) \right) \quad \forall k = 1, \ldots, n$$

where the λ_j parameters are determined to verify the constraints (and are thus linked to F_k) and Z is a normalizing function (a.k.a. partition function), i.e.,

$$Z = \sum_{i=1}^{n} \exp\left(\sum_{j=1}^{m} \lambda_j f_j(x_i) \right)$$

Feature functions in the translation example

$$P[Y = y|X = x; \Lambda] = \frac{1}{Z(x;\Lambda)} \exp\left(\sum_{j=1}^{m} \lambda_j f_j(x,y)\right)$$

Classification based on

$$\widehat{y} = \arg\max_y P[Y = y|X = x; \Lambda] = \arg\max_y \sum_{j=1}^{m} \lambda_j f_j(x,y)$$

In the translation example, feature functions are binary indicators, e.g.,

$$f_i(x,y) = \begin{cases} 1 & \text{if } y = \textit{en} \text{ and } \textit{April} \text{ follows } \textit{in} \\ 0 & \text{otherwise} \end{cases},$$

and we want $E[f(x,y)]$, i.e., the proba. of observing this configuration, to be equal to the empirical proportion of the configuration in the training data, e.g.,

$$E[f_i(x,y)] = \sum_{x,y} f_i(x,y)\tilde{p}(x)p(y|x) = \sum_{x,y} f_i(x,y)\tilde{p}(x,y) = F_i$$

About estimation

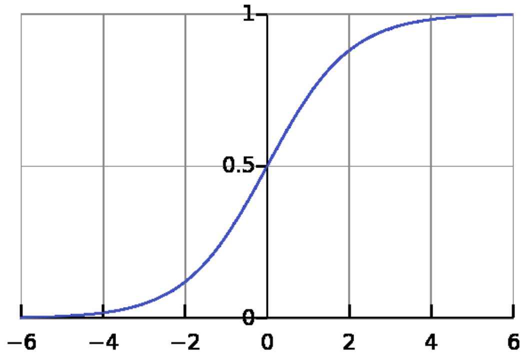
Parameters: $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$

Estimation: $\widehat{\Lambda}$ is the set of Lagrange multipliers obtained by the set of equations

$$F_k = \frac{\partial \ln Z(\Lambda)}{\partial \lambda_k}$$

Interestingly, $\widehat{\Lambda}$ will prove to maximize the conditional log-likelihood in classification problems (as we will see later).

Binary logistic regression



- Observed variables: $X = \{X_1, \ldots, X_n\}$
- Class variable: $Y \in \{0, 1\}$ (binomial distribution)
- We posit that the log ratio of posterior probabilities is a linear combination of the x_i, e.g.,

$$\ln \frac{P[Y=1|X]}{1-P[Y=1|X]} = \lambda_0 + \lambda_1 x_1 + \ldots + \lambda_n x_n$$

- The class posterior is given by

$$P[Y=1|X=x] = \frac{\exp\left(\lambda_0 + \lambda_1 x_1 + \ldots + \lambda_n x_n\right)}{1+\exp\left(\lambda_0 + \lambda_1 x_1 + \ldots + \lambda_n x_n\right)}$$

$$= \frac{1}{1+\exp\left(-\lambda_0 - \lambda_1 x_1 - \ldots - \lambda_n x_n\right)}$$

Multinomial logistic regression (Maxent)

- Observed variables: $X = \{X_1, \ldots, X_n\}$
- Class variable: $Y \in \{1, \ldots, K\}$ (multinomial distribution)

- Idea: One set of regression coefficient $\lambda_i^{(k)}$ for each class
- Class posterior probabilities are given by

$$P[Y = k | X = \mathbf{x}] = \frac{\exp(\lambda_0^{(k)} + \lambda_1^{(k)} x_1 + \ldots + \lambda_n^{(k)} x_n)}{\sum_i \exp(\lambda_0^{(i)} + \lambda_1^{(i)} x_1 + \ldots + \lambda_n^{(i)} x_n)}$$

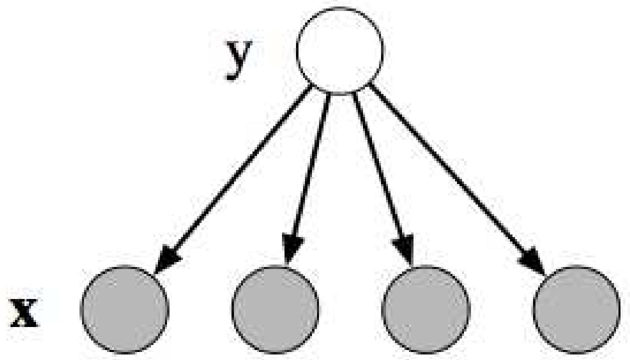
MAP and Maxent compared

MAP

$$p(y, \mathbf{x}) = p(y) \prod_{i=1}^{n} p(x_i|y)$$

MAXENT

$$p(y|\mathbf{x}) = \frac{1}{Z} \exp(b^{(k)} + \mathbf{w}^{(k)}\mathbf{x})$$



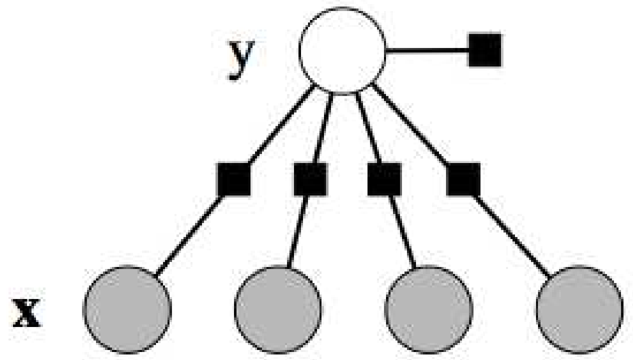
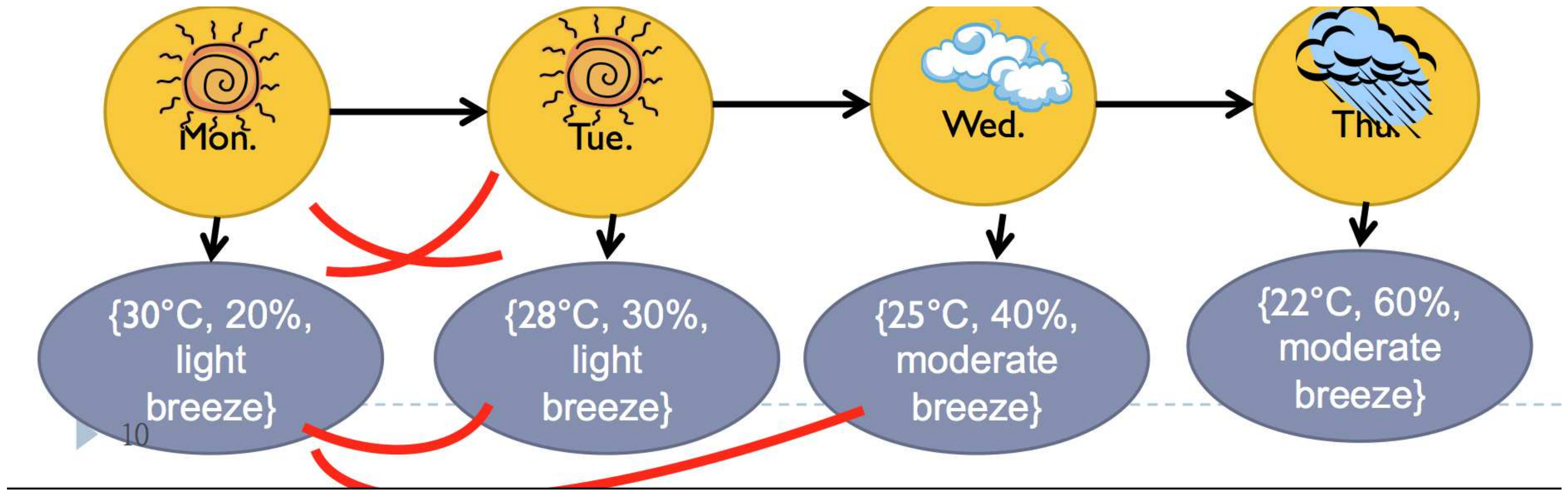


Figure taken from C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. Foundations and Trends in Machine Learning 4 (4). 2012

Log-linear models for sequences

X: sequence of n multivariate observations
Y: sequences of n labels



Mon.
Tue.
Wed.
Thu.
{30°C, 20%, light breeze}
{28°C, 30%, light breeze}
{25°C, 40%, moderate breeze}
{22°C, 60%, moderate breeze}
10

$$P[Y|X] = \frac{1}{Z(x,\Lambda)} \exp\left(\sum_{j=1}^{m} \lambda_j f_j(x,y)\right) \quad ?$$

Back to feature functions

- $f_j : X \times Y \to \mathbb{R}$ or $\{0, 1\}$ are usually specified by templates (class of feature functions) and are often of the form

$$f_j(x, y) = A_a(x) B_b(y) \ .$$

- $A_a(x)$ and $B_b(y)$ binary $\rightarrow$ binary conjunction indicator state/feature
- Feature functions are arbitrary and may overlap in any way

$A_1(x)$	$\mathbb{I}(x$ starts with a capital letter)
$A_2(x)$	$\mathbb{I}(x$ starts with letter G)
$A_3(x)$	$\mathbb{I}(x$ has 6 letters)

$\Rightarrow$ need for sequential features and tractability!

Feature functions for sequences

Feature functions

- can be position-specific ... or sum over all positions in the sequence
- can look at one x_s ... but also at a set of x_s up to x globally
- **will depend on at most two neighboring tags** ... not on y globally

  $\Rightarrow$ similar to Markov property in HMMs

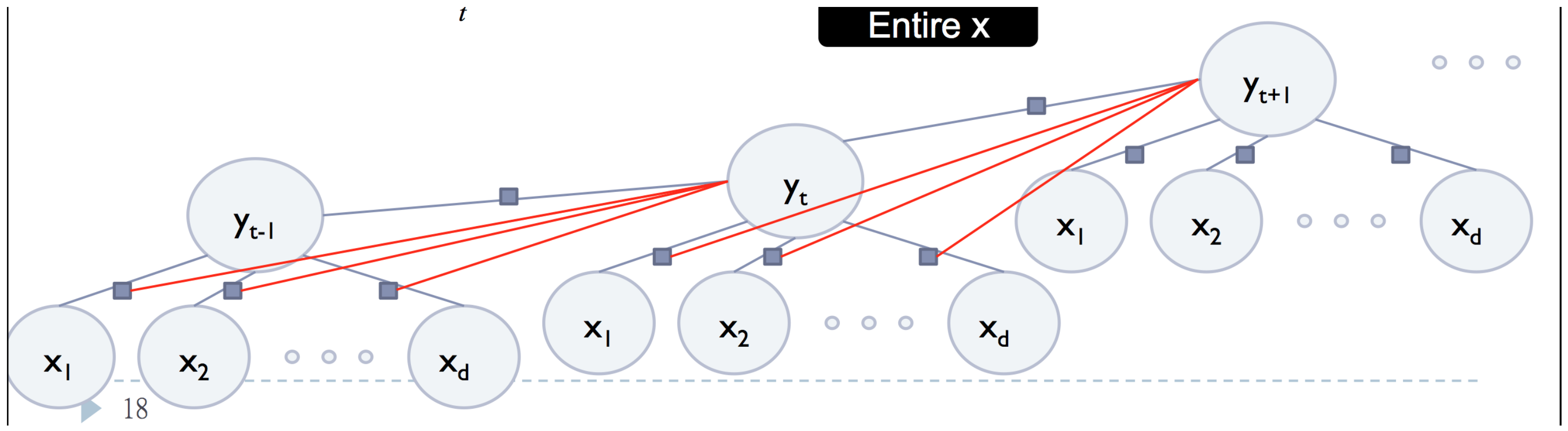
We will assume from now on (without loss of generality) that

$$f_i(x, y) = \sum_{s=1}^{n} f_i(y_{s-1}, y_s, x, s)$$

Advantages:

- fixed set of feature functions regardless of the sequence length
- tractable decoding (a.k.a. inference) as will be illustrated

Feature functions for sequences (cont'd)



POS tagging examples for $f(y_{s-1}, y_s, x, s)$:

y_s is NOUN and x_s is capitalized

x_{s-1} is 'Mr' and y_s is PROPER_NOUN

y_{s-1} is SALUTATION and y_s is PROPER_NOUN

Trick: y_0 = START and y_{n+1}=END

Feature functions for sequences (cont'd)

In standard toolkits, they are defined by (cryptic) *templates*, e.g., for BIO tagging on POS-tagged data

```
U00:%x[-2,0] # look at w_{k-2}
U01:%x[-1,0] # look at w_{k-1}
...
U17:%x[1,0]/%x[1,1] # look at w_{k+1} and pos_{k+1}
U18:%x[1,0]/%x[2,1] # look at w_{k+1} and pos_{k+2}
..
U20:%x[-2,1]/%x[-1,1]/%x[0,1]  # guess what????
```

```
The DT B-NP
pen NN I-NP
is VB B-VP
a DT B-NP
```

from which features are generated for each position in the data

The linear-chain CRF model

The posterior probability of a linear chain CRF is the Gibbs distribution given by

$$P[Y|X] = \frac{1}{Z(x,\Lambda)} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right)$$

where

$$f_i(x,y) = \sum_{s=1}^{n} f_i(y_{s-1}, y_s, x, s) \qquad \forall i \in [1,m]$$

The partition function is given by

$$Z(x,\Lambda) = \sum_{y} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right)$$

The parameters of the model are $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$.

Sequence decoding with linear-chain CRFs

Finding the state sequence which maximizes the posterior Gibbs distribution boils down to

$$\widehat{y} = \arg\max_y \frac{1}{Z(x,\Lambda)} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right) = \arg\max_y \sum_{i=1}^{m} \lambda_i f_i(x,y)$$

Since $f_i(x,y) = \displaystyle\sum_{s=1}^{n} f_i(y_{s-1}, y_s, x, s)$, we get

$$\begin{aligned} \widehat{y} &= \sum_{i=1}^{m} \lambda_i \sum_{s=1}^{n} f_i(y_{s-1}, y_s, x, s) \\ &= \sum_{s=1}^{n} \underbrace{\sum_{i=1}^{m} \lambda_i f_i(y_{s-1}, y_s, x, s)}_{g_s(y_{s-1}, y_s)} \end{aligned}$$

Dynamic programming with linear-chains CRFs

$H(k, v)$: score of the best sequence of tags y_1 to $y_k = v$

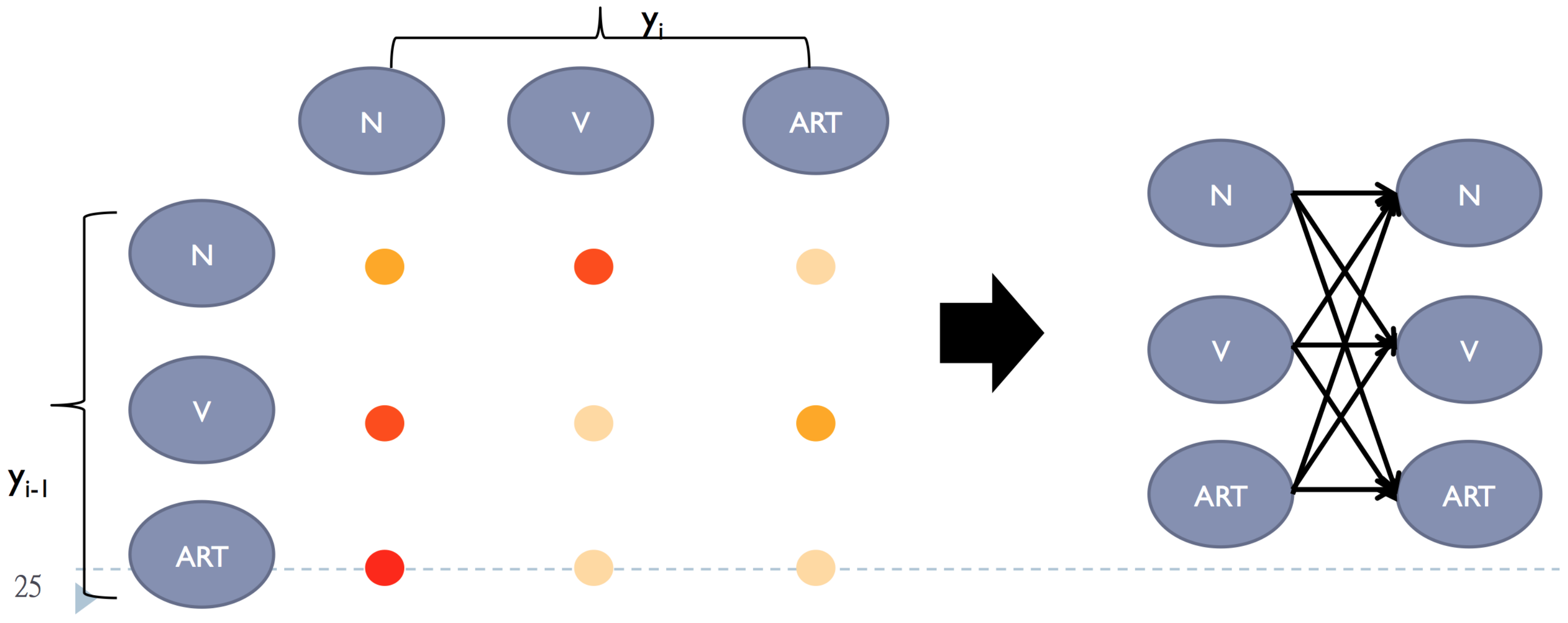
$$\begin{aligned} H(k, v) &= \max_{y_1,\dots,y_{k-1}} \sum_{s=1}^{k-1} g_s(y_{s-1}, y_s) + g_k(y_{k-1}, v) \\ &= \max_{y_{k-1}} \underbrace{\max_{y_1,\dots,y_{k-2}} \sum_{s=1}^{k-2} g_s(y_{s-1}, y_s) + g_{k-1}(y_{k-2}, y_{k-1})}_{H(k-1, y_{k-1})} + g_k(y_{k-1}, v) \end{aligned}$$

We finally get the recursion

$$H(k, v) = \max_u H(k-1, u) + g_k(u, v)$$

Dynamic programming with linear-chains CRFs

In practice, we use a Viterbi-like dynamic programming exploiting the $K \times K$ matrix gathering $g_s(y_{s-1}, y_s)$ functions for all pairs of tags.

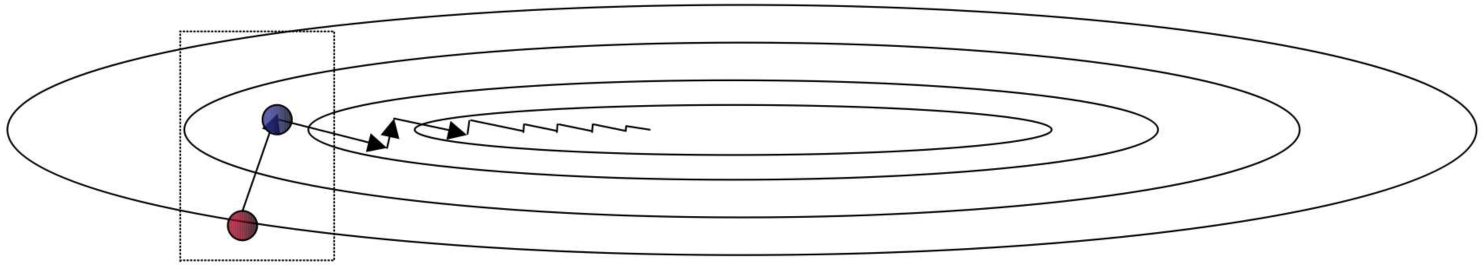


Parameter estimation

- **Objective**: Maximize $\ln p(y|x;\Lambda)$ on the training data $(x^{(r)}, y^{(r)})$, i.e.,

$$\widehat{\Lambda} = \arg\max_{\Lambda} \sum_r \left(\sum_i \lambda_i f_i(x^{(r)}, y^{(r)}) - \ln Z(x^{(r)};\Lambda) \right)$$

- no analytical solution (as usual with Gibbs distributions)
- $\cap$-convex objective function $\Rightarrow$ gradient ascent



Parameter estimation: Gradient computation

Limiting ourselves to a single training sample (x, y) *for sake of simplicity as of now*

$$\begin{aligned}
\frac{\partial}{\partial \lambda_i} \ln p(y|x;\Lambda) &= f_i(x,y) - \frac{\partial}{\partial \lambda_i} \ln Z(x,\Lambda) \\
&= f_i(x,y) - \frac{1}{Z(x,\Lambda)} \frac{\partial}{\partial \lambda_i} Z(x,\Lambda)
\end{aligned}$$

So we need to compute $\frac{\partial}{\partial \lambda_i} Z(x,\Lambda)$.

Parameter estimation: Gradient computation (cont'd)

$$
\begin{aligned}
\frac{\partial}{\partial \lambda_i} Z(x,\Lambda) &= \sum_y \frac{\partial}{\partial \lambda_i} \exp\left(\sum_j \lambda_j f_j(x,y)\right) \\
&= \sum_y f_i(x,y) \exp\left(\sum_j \lambda_j f_j(x,y)\right)
\end{aligned}
$$

Hence

$$
\frac{\partial}{\partial \lambda_i} \ln p(y|x;\Lambda) = f_i(x,y) - \frac{1}{Z(x,\Lambda)} \sum_{y'} f_i(x,y') \exp\left(\sum_j \lambda_j f_j(x,y')\right)
$$

Log conditional likelihood and Maxent

Taking the partial derivative of the log conditional likelihood over the entire training set and setting to zero, we have

$$\underbrace{\sum_r f_i(x^{(r)}, y^{(r)})}_{\text{empirical expectation}} \quad = \quad \underbrace{\sum_r E[f_i(x,y)|x_{(r)}; \widehat{\Lambda}]}_{\text{theoretical expectation}}$$

CAUTION
WATCH
YOUR STEP

Equality holds for the entire training data, not for a single point!

Linear-chain CRF training (at last)

Foreach training couple $(x^{(r)}, y^{(r)})$, do

1. compute (or estimate) $E[f_i(x,y)|x^{(r)};\Lambda]$ given a current estimate of the parameters
   $\rightarrow$ done (somewhat) efficiently with a forward-backward algorithm
   $\rightarrow$ alternative solutions are Gibbs sampling, iterative scaling
2. compute $f_i(x^{(r)}, y^{(r)})$
3. update parameters using grandient ascent, i.e.,

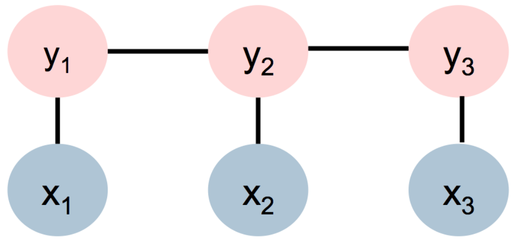
$$\lambda_i \leftarrow \lambda_i + \alpha \left(f_i(x^{(r)}, y^{(r)}) - E[f_i(x,y)|x_{(r)};\Lambda] \right)$$

CAUTION WATCH YOUR STEP

A bit more complex in practice! Need for regularization, other gradient optimization methods (e.g., BFGS).

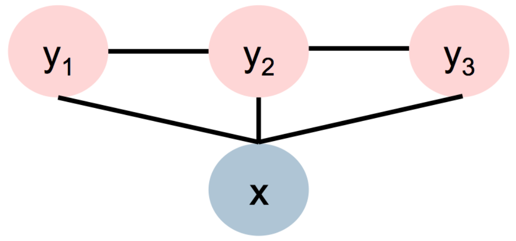
Linear-chain CRF *vs.* HMM

$$P[X = x_1, \ldots, x_T] = \exp\left(\ln \pi_{x_1} + \sum_{t=2}^{T} \ln a(x_{t-1}, x_t)\right)$$



$$\Downarrow$$

$$g(x_{t-1}, x_t) = \sum_{i,j} \xi_{ij}\, \delta(x_{t-1} = i)\, \delta(x_t = j)$$



Linear-chain CRF *vs.* HMM (cont'd)

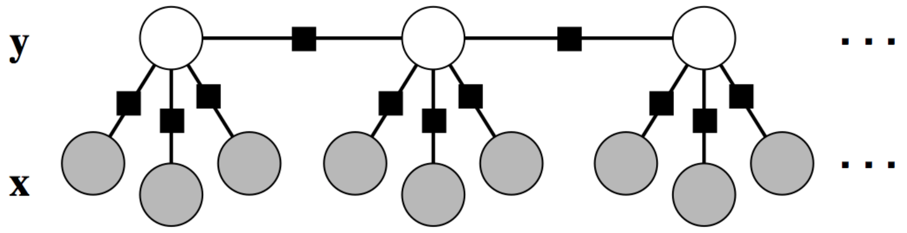


Figure 1.3 Graphical model of an HMM-like linear-chain CRF.

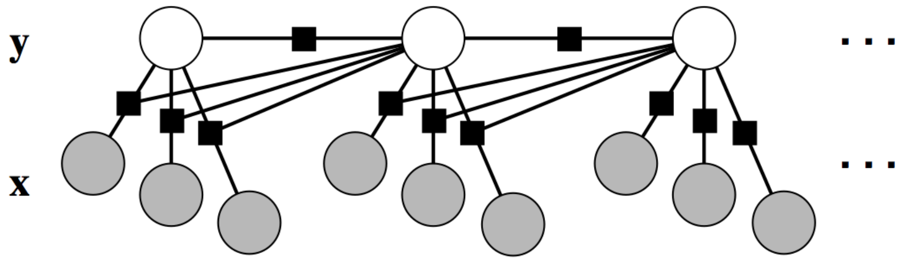
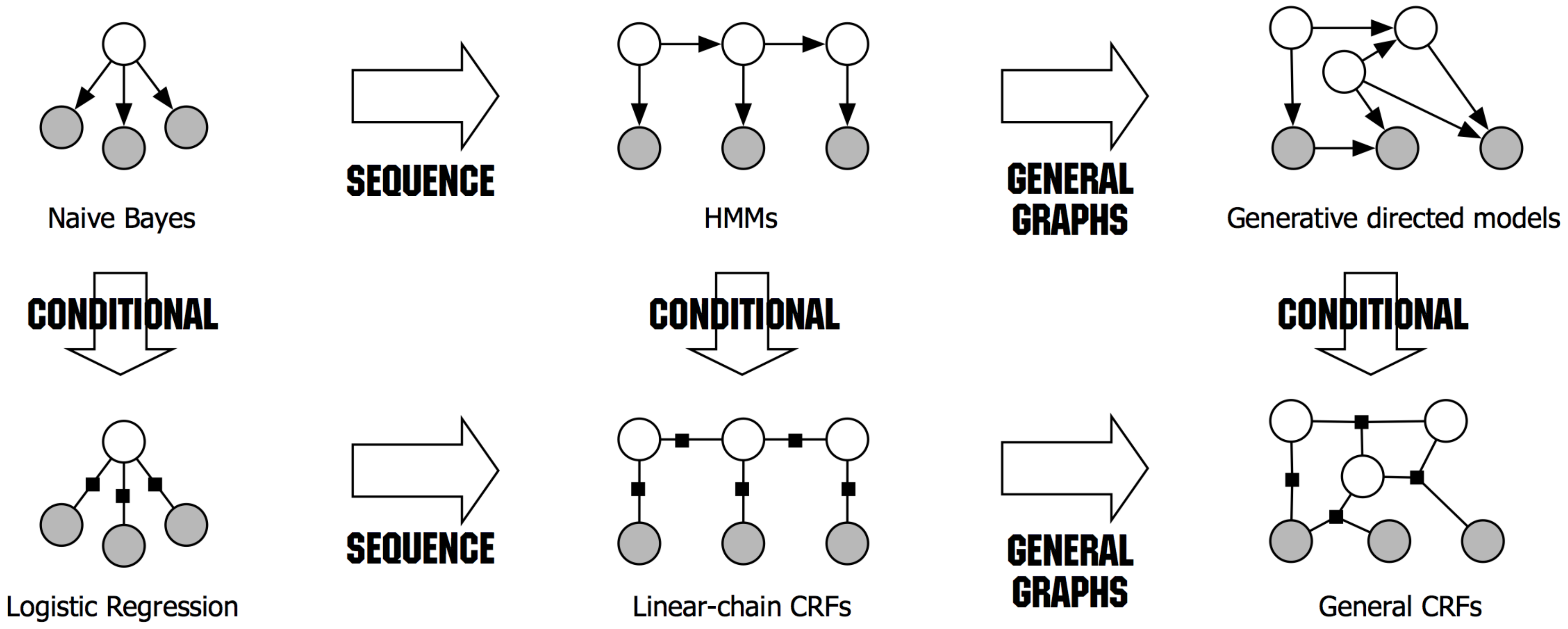


Figure 1.4 Graphical model of a linear-chain CRF in which the transition score depends on the current observation.

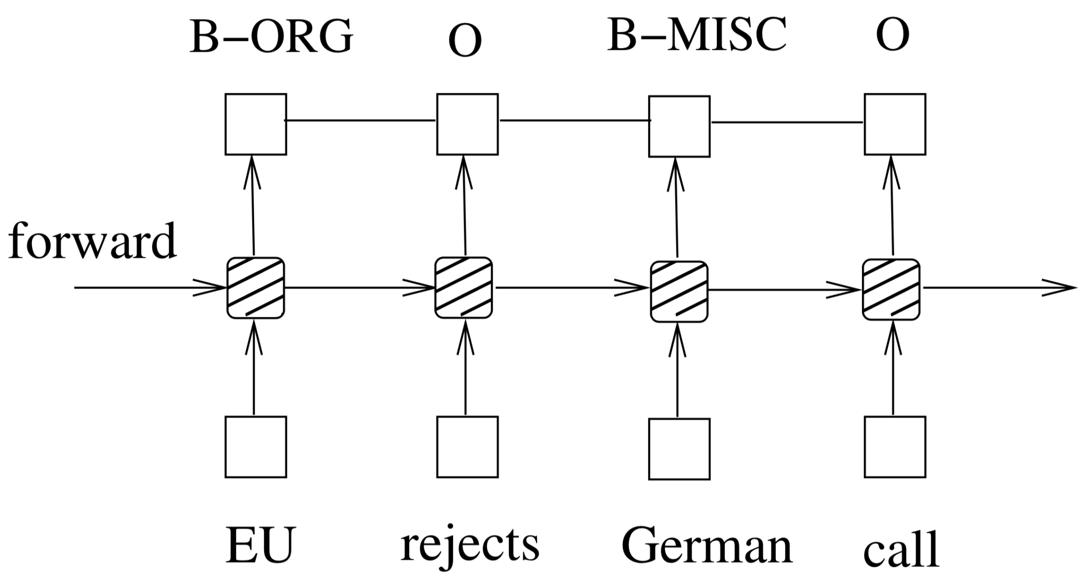
[From C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. Foundations and Trends in Machine Learning 4 (4), 2012]

Beyond HMMs and linear-chain CRFs



Tractability rapidly becomes an issue!

Mixing LSTMs and CRFs



LSTM directly predicts y_i from h_i

$$L(y_1, \ldots, y_n) \propto \sum_{k=1}^{n} \underbrace{\text{softmax}(f(x_k, h_k))[y_k]}_{P(k,x,y_k) \text{ = probability for tag } y_k \text{ at position } k}$$

CRF layer adds dependency between neighboring tags

$$L(y_1, \ldots, y_n) \propto \sum_{k=1}^{n} \underbrace{A(y_{k-1}, y_k)}_{\text{transition}} + P(k, x, y_k)$$

Mixing LSTMs and CRFs (cont'd)

Formally, from the score $s(y,x) = \sum_k A(y_{k-1}, y_k) + P(k, x, y_k)$, we define the posterior probability

$$p(y|x) = \frac{\exp s(y,x)}{\sum\limits_{y' \in \mathcal{Y}} \exp s(y', x)}$$

Decoding maximizes $\ln(p(t|w))$ in two steps

1. run LSTM to generate tag probability distribution $P(k, :)$ for all k
2. dynamic programming $P(k, x,)$

Back-propagation follows the same two step procedure

$s(y,x)$ can be seen as the potential function in CRF models, equivalent to $\sum_i \lambda_i f_i(x,y)$ for complex feature functions $f_i()$

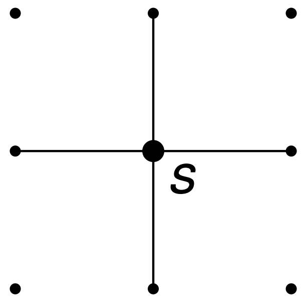
Zhiheng Huang et al., 2015. Bidirectional LSTM-CRF models for sequence tagging.

Markov random fields

Markov property in random fields

$$P[X_s = x_s | X_{S\setminus s} = x_{S\setminus s}] = P[X_s = x_s | X_{V_s} = x_{V_s}]$$

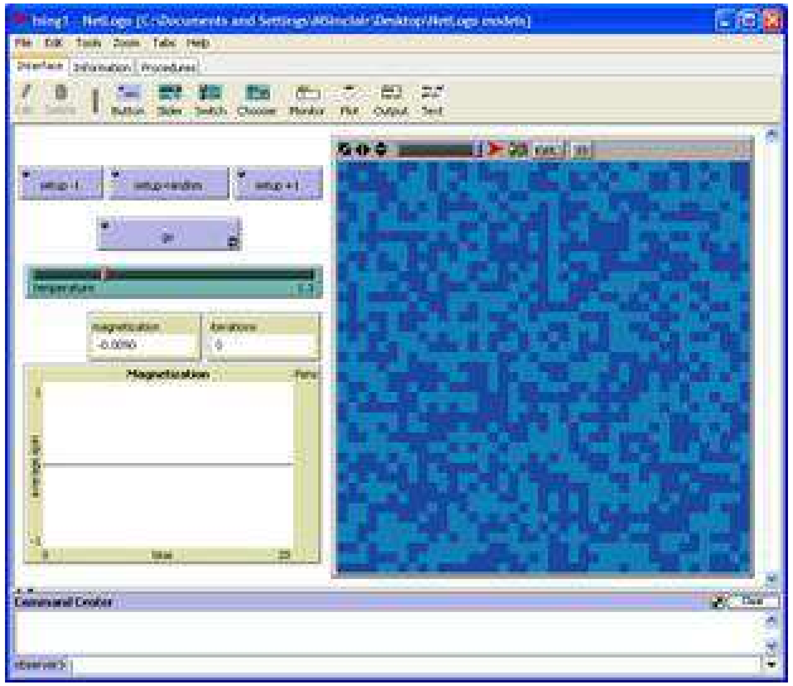
- $\forall s \in S \quad x_s \in E$
- sample space: $\Omega = E^{|S|}$
- neighborhood system: V
- set of cliques : $c \in \mathcal{C}$

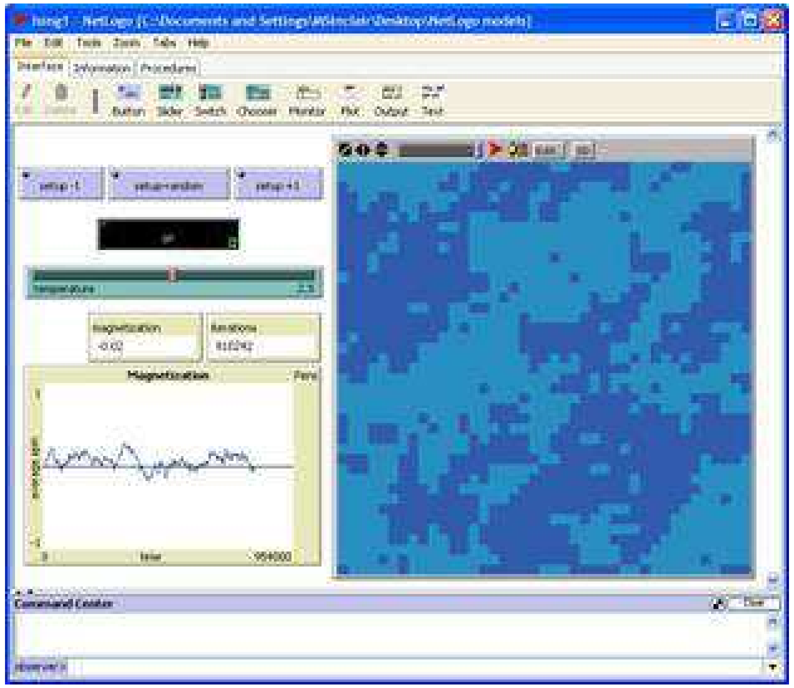


$$P[X = x] = \frac{1}{Z} \exp - \underbrace{\sum_{c \in \mathcal{C}} U_c(x)}_{U(x)} \qquad \text{with } Z = \sum_{x \in \Omega} \exp -U(x)$$

Markov random fields: an example

$$P[X_s|X_{V_s}] = \frac{\exp\left(\alpha x_s + \sum_{r\in V_s} \beta x_s x_r\right)}{1+\exp\left(\alpha x_s + \sum_{r\in V_s} \beta x_s x_r\right)}$$





Hidden Markov random fields

$X \longmapsto Y$, version *bruitée* du champ X

indépendance conditionnelle des données

$$\Downarrow$$

$$P[Y = y|X = x] = \prod_{s \in S} b_{x_s}(y_s) = \exp - \underbrace{\sum_{s \in S} - \ln b_{x_s}(y_s)}_{U(y|x)}$$

$P[X = x|Y = y]$: distribution de Gibbs d'énergie $U(x) + U(y|x)$