Natural Language Processing
CSE 325/425

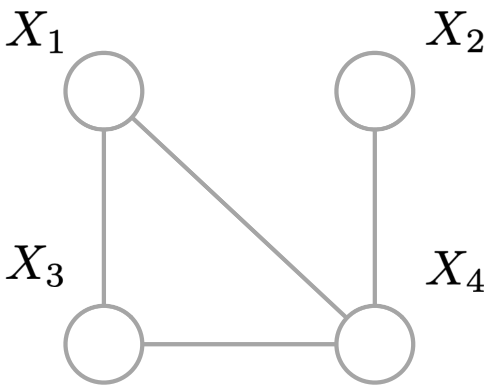
Sihong Xie

Lecture 11:
- Conditional random field
- Neural network revisit (forward propagation)

Graphical models

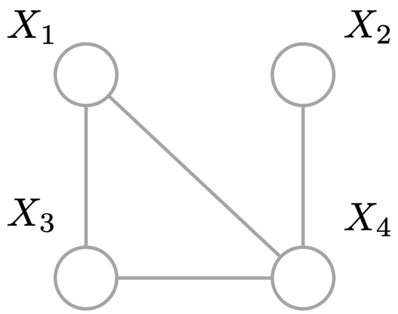
Conditional independence in a graphical model:

- $A \perp\!\!\!\perp B | C$ if any path from A to B have to pass some variables in C.



Cliques

Cliques and maximal cliques



- A clique ***c*** is a set of nodes that are fully connected
  - any two nodes in the clique are connected by an edge.
  - two nodes not in a clique **can** become conditional independent.

$$\Pr(X_i, X_j | X_{\setminus i,j}) = \Pr(X_i | X_{\setminus i,j}) \Pr(X_j | X_{\setminus i,j})$$

- A maximal clique ***c*** is a clique that adding any new node will make their union not a clique.
  - the collection of maximal cliques on a graphical model encode all conditional independence properties.

Factorization

Factorization using cliques.

$$\Pr(X_1, \ldots, X_n) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(X_c)$$

where

- ***c*** is a clique.
- $\psi_c(X_c) \geq 0$ is a potential function of the variable in the clique ***c***.
  - this is not a joint distribution of the variables X_c
- The normalization factor is defined as

$$Z = \sum_{x_1, \ldots, x_n} \prod_{c \in \mathcal{C}} \psi_c(X_c)$$

Conditional random fields

Conditional random fields:

- A “random field” refer to “a set of dependent random variables”.
  - a specific type of “graphical models”.
- “Conditional” means “conditioning on observed data”
  - making CRF a discriminative model (vs. generative models such as HMM).
- Use the maximum entropy principle – a generalization of logistic regression.
  - each factor is in the form
    - ***c*** is a maximum clique

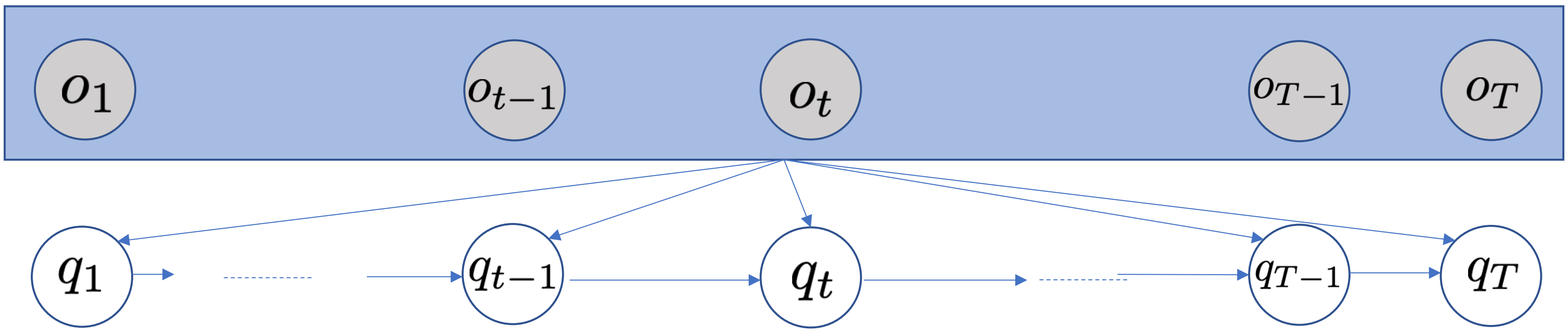
$$\psi_c(X_c; O) = \exp\left\{\sum_{i=1}^{d} \theta_i f_i(X_c; O)\right\}$$

  - joint conditional distribution

$$\Pr(X_1, \ldots, X_n; O) = \frac{1}{Z(O)} \prod_{c \in \mathcal{C}} \exp\left\{\sum_{i=1}^{d} \theta_i f_i(X_c; O)\right\}$$

CRF for POS tagging

The graphical model is a linear chain



All factors (or equivalently, maximum cliques) are pairwise

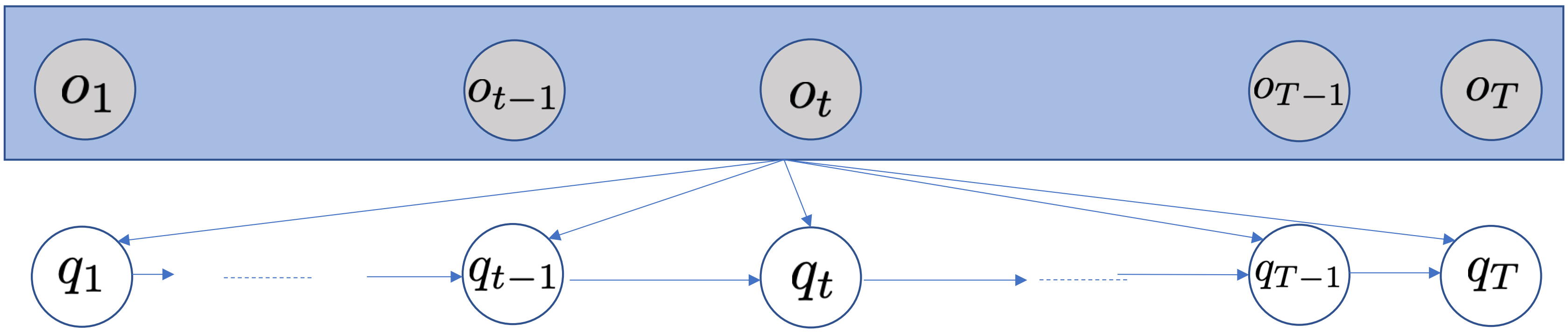
$$\psi_{t-1,t}(q_t, q_{t-1}; O) = \exp\left\{\sum_{i=1}^{d} \theta_i f_i(q_t, q_{t-1}; O)\right\}$$

The inner product = the compatibility score of the two tags:

$$s_t(q_{t-1}, q_t; \boldsymbol{\theta}, O) = \langle \boldsymbol{\theta}, \mathbf{f}(q_{t-1}, q_t) \rangle$$

CRF for POS tagging

The graphical model is a linear chain



- The joint distribution of a tag sequence, conditioned on a word sequence is

$$\Pr(q_1, \ldots, q_T; O) = \frac{1}{Z(O)} \prod_{t=1}^{T} \exp\left\{ \sum_{i=1}^{d} \theta_i f_i(q_t, q_{t-1}; O) \right\}$$

decompose over steps of the sequence:
=> conditional independence
=> polynomial inference alg.

$$= \frac{1}{Z(O)} \exp\left\{ \sum_{t=1}^{T} \sum_{i=1}^{d} \theta_i f_i(q_t, q_{t-1}; O) \right\} = \frac{1}{Z(O)} \exp\left\{ \sum_{t=1}^{T} s_t(q_{t-1}, q_t; \boldsymbol{\theta}, O) \right\}$$

Predicting sequence using CRF

Input:

- an input sentence $O = [o_1, \ldots, o_T], o_t \in V$
- and a trained CRF model $\boldsymbol{\theta}$

Output:

- optimal POS tag sequence $$Q^* = \arg\max_Q \Pr(Q|O;\boldsymbol{\theta}) = \arg\max_Q \sum_{t=1}^{T} s_t(q_{t-1}, q_t; \boldsymbol{\theta}, O)$$

Adapt the Viterbi algorithm for HMM to CRF prediction:

- change the scores in HMM $s_t(q_{t-1} = i, q_t = j; \lambda, O) = a_{i,j} b_j(o_t)$

  to the scores defined for CRF.

Learning a CRF model

Not much more difficult than training a logistic regression model!

- Input: m POS-tagged sentences.
- MLE: $$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log \Pr(Q^{(i)}|O^{(i)}, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \ell(Q^{(i)}|O^{(i)}, \boldsymbol{\theta})$$
- There is no closed form solution for the parameter, and gradient descent is needed.

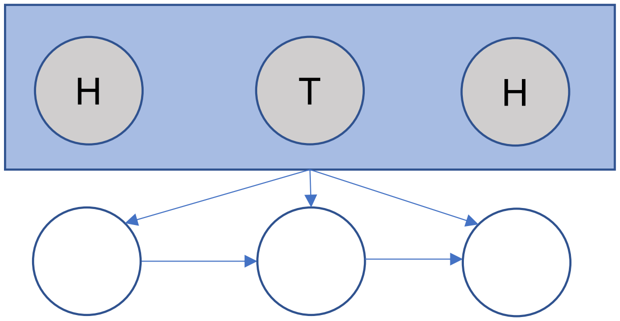
$$\frac{\partial}{\partial \boldsymbol{\theta}} \ell(Q|O, \boldsymbol{\theta})$$

decompose over steps of the sequence:
=> conditional independence
=> polynomial inference alg.

- Recall the gradient for multi-class logistic regression …

Learning a CRF model

A running example (Cheating Casino)



Revisiting neural networks

POS tagging goes neural!

- use a neural network to predict sequences.
- upgrade HMM/MEMM to RNN
- upgrade CRF to CRF-LSTM

Matrix Calculus

- Usually we minimize a scalar loss with respect to a set of parameters organized in a bunch of vectors.

$$\min_{w \in \mathbb{R}^d} L(w)$$

- Most optimization algorithms need the gradient of the loss with respect to vectors.

$$\frac{\partial L}{\partial w}$$

- Logistic regression optimization gives a simple case.

Matrix Calculus

- Basic definitions. Given a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$
  - differential $f(x+h) = f(x) + d_x f(h) + o(h)$ $\quad d_x f : \mathbb{R}^n \to \mathbb{R}$
  - gradient: explicit form of $d_x f$ $\quad \nabla_x f : \mathbb{R}^n \to \mathbb{R}^n$
  - partial derivative

$$\nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

  - Examples:

$$f([x_1, x_2]) = 3x_1 + x_2^2$$

Matrix Calculus

- Generalization of gradient to higher dimensional output space $f : \mathbb{R}^n \to \mathbb{R}^m$

  - Jacobian

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

  - Examples:

$$g([y_1, y_2, y_3]) = \begin{bmatrix} y_1 + 2y_2 + 3y_3 \\ y_1 y_2 y_3 \end{bmatrix}$$

Matrix Calculus

- Chain rule
  - Given two differentiable functions $g : \mathbb{R}^p \to \mathbb{R}^n \quad f : \mathbb{R}^n \to \mathbb{R}^m$
  - Composition of two functions $f \circ g : \mathbb{R}^p \to \mathbb{R}^m$
  - Jacobian of $f \circ g$ is the product of the Jacobians

$$d_x(f \circ g) = d_{g(x)}(f) \circ d_x(g) \qquad J_{f \circ g} = J_f \circ J_g$$

  - Examples:

$$g([y_1, y_2, y_3]) = \begin{bmatrix} y_1 + 2y_2 + 3y_3 \\ y_1 y_2 y_3 \end{bmatrix}$$

$$f([x_1, x_2]) = 3x_1 + x_2^2$$

Matrix Calculus

- Common examples
  - Matrix-vector (wrt vector) $$\frac{\partial}{\partial x} W x = W$$
  - Vector-matrix (wrt vector) $$\frac{\partial}{\partial x} x^\top W = W^\top$$
  - Matrix-vector (wrt matrix) $$\frac{\partial}{\partial W} W x$$
    - Usually don't directly find this Jacobian.
    - Rather, embed this in chain-rule for $J = f(z) \qquad z = Wx$
    - Example in the next slide.

Matrix Calculus

- Common examples
  - Cross-entropy loss (negative log-likelihood loss)
  - Used in logistic regression and neural networks

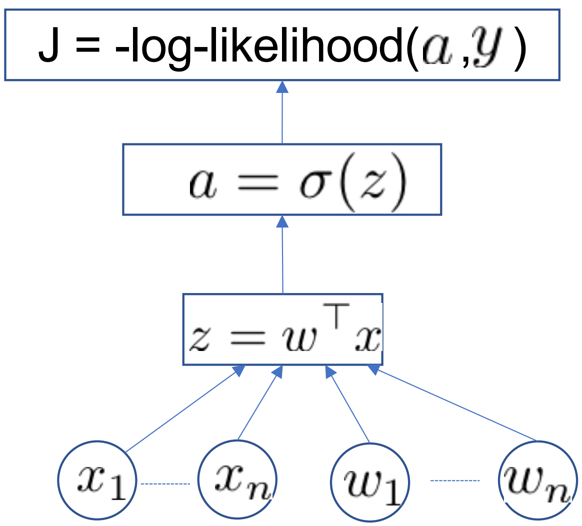
$$z = Wx$$

$$\hat{y} = \mathrm{softmax}(z)$$

$$CE(y, \hat{y}) = -\sum_i y_i \log \hat{y}_i$$

Logistic regression is a neural network

A computation graph is a differentiable system for ***evaluation*** and ***differentiation***.



Forward pass:

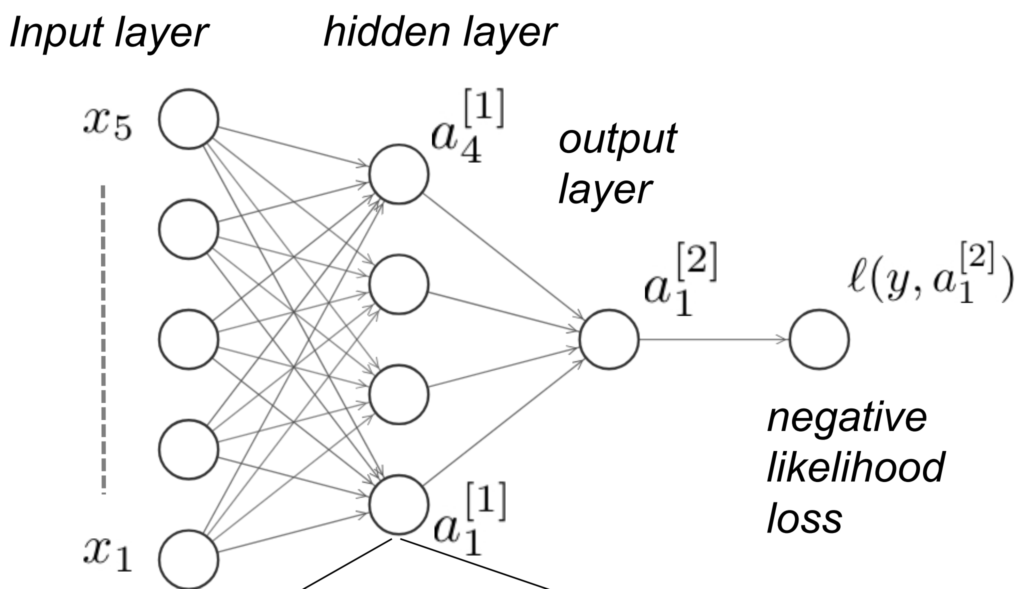
- Compute the value of the hidden, output units, and the loss.

Back-propagation:

- compute the gradients using the chain rules.

Neural Network

A neural network is a computation graph that stacks many Logistic regression models.



One Logistic regression model

$z_1^{[1]} = {W_1^{[1]}}^\top x \qquad a_1^{[1]} = \sigma(z_1^{[1]})$

$W_1^{[1]}$ the vector of weights on edges connecting x to $z_1^{[1]}$

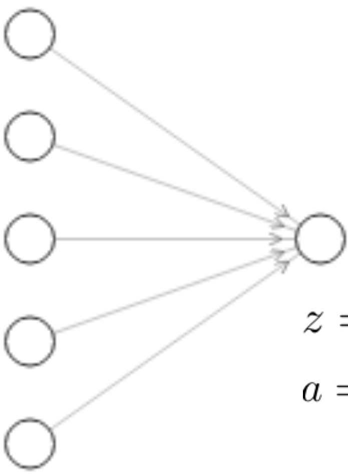
Neural network drawing tool: http://alexlenail.me/NN-SVG/index.html

Forward propagation

In general, for the j-th neural on the first layer:

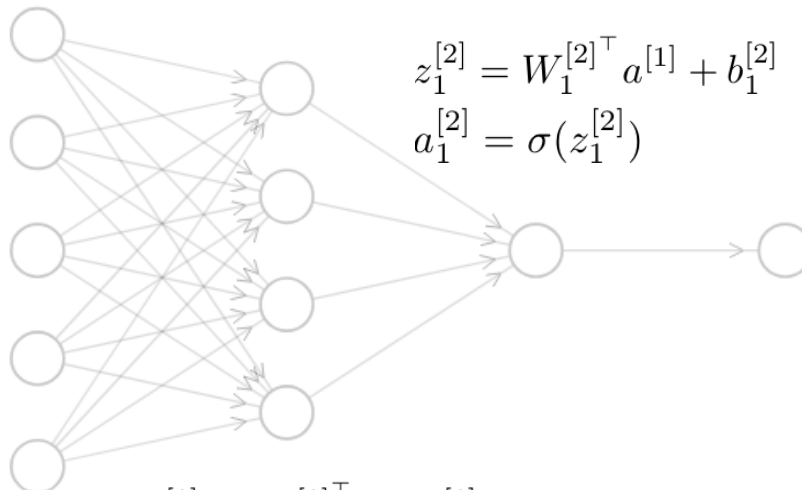
$$z_j^{[1]} = W_j^{[1]^\top} x + b_j^{[1]}$$

$$a_j^{[1]} = \sigma(z_j^{[1]})$$



$$z = w^\top x + b$$

$$a = \sigma(z)$$



$$z_1^{[2]} = W_1^{[2]^\top} a^{[1]} + b_1^{[2]}$$

$$a_1^{[2]} = \sigma(z_1^{[2]})$$

$$z_1^{[1]} = W_1^{[1]^\top} x + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$