x → log → a → exp() → b | Reverse Mode Automatic Differentiation recursively applies Chain-Rule $\partial y_i/\partial x_j = \sum_{p \in Paths(i,j)} \prod \partial z_v/\partial z_k$ ⇒ exponential paths $O(2^n)$ or fully connected $O(Neurons^{\#Hiddenlayer})$

Backprop Constructive theorem: Same asymptotic complexity as original function, many inefficient algos pwl.

$f'(x) = \lim_{h\to 0} (f(x+h)-f(x))/h$ $O(h^2)$ | $\frac{\partial}{\partial x} f(g(x)) = f'(g(x)) \cdot g'(x)$ | $\theta^{new} = \theta^{old} - \gamma \cdot \partial/\partial\theta \cdot \sum L(f(x;\theta^{old}), y)$

$c=0$ | $x=1$ | $ax=a$ | $x^2=2x$ | $\sqrt{x} = (1/2)\cdot x^{-1/2}$ | $e^x = e^x$ | $a^x = \ln(a)\cdot a^x$ | $\ln(x) = 1/x$ | $\sin = \cos$ | $\cos = -\sin$

Log-linear model very general family of distributions, exponential of lin. function: $p(y|x,\theta) = \exp(\theta\cdot f(x,y)) / \sum_{y'\in Y} \exp(\theta\cdot f(x,y'))$

How good is x/y together ≥0 | Partition function normalizes | $\log(p(y|x,\theta)) = \theta\cdot f(x,y) + const$

$\theta^* = \arg\max L(\theta) = \arg\max \prod_{n=1}^{N} p(y_n|x_n,\theta) = \arg\min -\sum_{i=1}^{N} \log p(y_n|x_n,\theta)$ → $\nabla^2 LL(\theta) = \hat{\Sigma}_{i=1}^{n}(cov(f(x_i,y)))$ (covariance matrix of random vector $f(x_i,y)$)

$\frac{\partial LL(\theta)}{\partial\theta_k} = -\sum_{i=1}^{n} \frac{\partial \log(p(y_i|x_i,\theta))}{\partial\theta_k} = -\sum_{i=1}^{n} \frac{\partial}{\partial\theta_k}(\theta\cdot f(x,y) - \log[\sum_{y'\in Y}\exp(\theta\cdot f(x,y'))]) = -\sum(f_k(x_i,y_i) - \frac{\sum_{y'\in Y}\exp[\theta\cdot f(x_i,y')]\cdot f_k(x_i,y')}{\sum_{y'\in Y}\exp(\theta\cdot f(x_i,y'))}) = -\sum(f_k(x_i,y_i) - \sum_{y'\in Y} p(y'|x_i,\theta)\cdot f_k(x_i,y'))$

$= -\sum f_k(x_i,y_i) + \sum_{i=1}^{n}\sum_{y'\in Y} p(y'|x_i,\theta) f_k(x_i,y') = -\sum_{i=1}^{n} f_k(x_i,y_i) + \sum_{i=1}^{n} \mathbb{E}_{y|x=x_i}[f_k(x_i,y)]$ (=Observed feature count = expected feature count when $\nabla LL(\theta)=0$)

Softmax $h_i, Y, T) = \exp(h_y/T) / \sum_{y'\in Y}\exp(h_{y'}/T)$ | Log-linear is dot product followed by softmax $h_y = \theta\cdot f(x,y)$

$\log(softmax(h_i,y)) = h_y - \log\sum_{y'\in Y}\exp(h_{y'})$ | $\frac{\partial \log(softl)}{\partial h_i} = \delta_{y_i} - (1/\sum_{y'\in Y}\exp(h_{y'}))\cdot\exp(h_i) = \delta_{y_i} - softmax(h,i)$ $(\delta_{ik}=0 \Rightarrow i\neq k$, $\delta_{ik}=1 \Rightarrow i=k)$

T = Temp. Parameter | $T\to\infty$: approach uniform categorical dist. (max entropy) ← Interpolate softmax → argmax

$T\to 0$: annealing, all mass placed on maximum (min entropy) | leading argmax as dist.

Exp. Family: specific dist. varies with parameter | $p(x|\theta) = 1/z(\theta)\, h(x)\cdot\exp(\theta\cdot\phi(x))$

Sigmoid: $\sigma(x) = \exp(x)/(1+\exp(x)) = 1/(1+\exp(-x)) = 1/(1+e^{-x}) \in (0,1)$ | Sigmoid is binary softmax

We can hack non-linear feature function into log-linear, but we need to know function ⇒ Neural Net.

Learn parameters of $softmax(\theta\cdot f(x,y))$ f and θ: $h^2 = \sigma^2(W^2\cdot\sigma^1(W^1\cdot x + b^1) + b^2) = softmax(h^n, y)$ ↙(-1,1)

Function Engineering: $f(x,y) = [x_1, x_1^2, x_2, x_2^2, -1]$ | Non linear activation function: $\tanh(x) = 2\sigma(2x)-1$ / relu

RNN: has time dependant hidden state, language model with infinite context | Cell: $h_t = f(h_{t-1}, e(y_{t-1}))$

Vanilla RNN: Elman Networks: incoporate arbitrary distant contextual information & treat word prediction as a discriminative learning task: $p(y_t|y_{<t}) = p(y_t|y_1,\dots,y_{t-1}) = \exp(w_{y_t}\cdot h_t)/\sum_{y\in V}\exp(w_{y'}\cdot h_t)$ h_{t-1} → h_t → tanh → y_t, x_t

Vanila RNN: $h_t = \sigma(W_1 h_{t-1} + W_2\cdot e(y_{t-1}))$ (=combination of hidden + current state)

↑ Non-linearity e.g. tanh | Instead of fixed context considered at every step ⇒ ∞ context

Linguistic Phenomena: long distance dependencies: verb-subject agreement in German subordinate clauses ↙

Language Modeling is structured prediction where we have strings, trees, graphs. It is the intersection of algos. and high-dim. statistics & is weighting of prefix-tree | $y\in Y$ is extremly large 2^n

Voc V finite, Kleene Closure V^* infinite (infinite number of paths with finite length)

Local Normalization: Choose weights of syntax-tree such that $\sum_{y\in V^*}\prod_{t=1}^{|y|} p_{y'}(y_t) = 1 \Rightarrow Z = 1$

You multiply weights along a path and then sum all paths

N-gram assumption: condition only n-1 words: $p(w) = p(w_1)\cdot p(w_2|w_1)\cdot\dots\cdot p(w_k|w_1,\dots,w_{k-1}) \overset{n-1}{\Leftrightarrow} p(w_i|w_1,\dots,w_{i-1}) = p(w_i|w_{i-n+1},\dots,w_{i-1})$

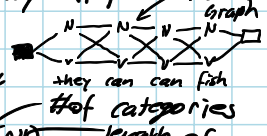
Bengio neural model: problem: discrete random variables: 10 consecutive words, $|V| = 100'000 \Rightarrow 100'000^{10}-1$ params.

↳ Pass word embedding to MLP and reduce parameters to generalize better ← feature function

1. Word embedding like skip gram → e(history) | 2. MLP combines embeddings $h_t = f(e(history))$ | 3. softmax for prob.

Recurrent Neural Language Model: $h_t = f(h_{t-1}, e(y_{t-1}))$ | $p(y_t|y_{<t}) = \exp(w_{y_t}\cdot h_t)/\sum_{y'\in V}\exp(w_{y'}\cdot h_t)$

Part-of-speech-tagging (=POS): Determiner (the), Noun, verb, etc. is like path search in Graph



Score(<grammar tag sequence>, sentence) | CRF: conditional probabilistic model for sequence labeling

↳ higher better: can be any function or NN · built on logistic regression classifier (log-lin model)

$p(t|w) = \exp(score(t,w))/\sum_{t\in T}\exp(score(t',w))$ | Problem: exp. number of POS assignments $O(|T|^{N})$

Use combinatorial structure of sentence: $score(t,w) = \sum_{n=1}^{N} score(<t_{n-1},t_n>, w)$ (=Every arc gets own score)

Normalizer: $\sum_{t\in T^N}\exp(\sum_{n=1}^{N} score(<t_{n-1},t_n>, w)) = \sum_{t_1\in T}\exp(score(<t_0,t_1>, w))\cdot\dots\cdot\sum_{t_N\in T}\exp(score(<t_{N-1},t_N>, w))$

Calculate normalizing constant $Z = \beta(w,t_0)$: 1. $\beta(w,t_N) \leftarrow 1$ | 2. for $n \leftarrow N-1,\dots,0$: (=Like backpropagation)

3. $\beta(w,t_n) \leftarrow \sum_{t_{n+1}\in T}\exp(score(<t_n,t_{n+1}>, w))\cdot\beta(w,t_{n+1})$ (=Backward algorithm)

Compute max path: Viterbi: 1. $\gamma(w,t_N) \leftarrow 1$ | 2. for $n\in N-1,\dots,0$: | 3. $\gamma(w,t_n) \leftarrow \max_{t_{n+1}\in T}\exp(score(t_n,t_{n+1}))\cdot\gamma(w,t_{n+1})$

Semiring: $R = <\mathbb{R}_+, +, \times, 0, 1>$: $(a+b)+c = a+(b+c)$ | $0+a = a+0 = a$ | $a+b = b+a$

$(a\times b)\times c = a\times(b\times c)$ | $1\times a = a\times 1 = a$ | $a\times(b+c) = (a\times b)+(a\times c)$ | $(a+b)\times c = (a\times c)+(b\times c)$ | $0\times a = a\times 0 = 0$

Viterbi: $[0,1]$, max, x, 0, 1 | Inside: $\mathbb{R}^+ \cup \{+\infty\}$, +, x, 0, 1 ← Semiring $R = (\mathbb{R}^+, +, \times, 0, 1)$

Normalizer: $Z = \beta(w,t_0)$: 1. $\beta(w,t_N) = 1$ | 2. for $n = N-1,\dots,0$: | 3. $\beta(w,t_n) = \bigoplus_{t_{n+1}\in T}\exp(score(t_n,t_{n+1}))\otimes\beta(w,t_{n+1})$

Highest Path: Same as above but with semiring $R = <\mathbb{R}^+, \max, \times, 0, 1>$ ← max. this log-likelihood it's softmax

Parameters for score with backprop: max. log-likelihood: $\sum_{i=1}^{I} score(t^i, w^i) - \log\sum_{t'\in T^N}\exp(score(t', w^i))$

Constituent: word/group that function as single unit: [fruit flies] [like [a green banana]]

Context Free Grammar: rules can be applied to nonterminal regardless of context | grammar: set of rules

$G = <N$: non terminal symbols $N_1,\dots$, S: start S, Σ: alphabet of terminals a_1, R: rules $N\to\alpha > NP\,VP,\dots$

Probabilistic CFGs: prob. to each rule | multiply all probs. in tree to get prob. of tree ↙ ↙ S →

Locally normalized: prob. in rule sums to 1: $S\to NP\,VP$ (0.6) | $N\,VP$ (0.4) ⇒ $\sum_{k=1}^{K} p(\alpha_k|N) = 1$

Weighted CFGs: globally normalized: $\exp(score(Tree))$: just structured softmax: $p(t) = \frac{1}{Z}\prod_{r\in tree}\exp(score(rule))$

Parsing: create tree from sentence | Chomsky Normal Form (CNF): $N_1\to N_2 N_3$, $N\to a$ ↖ Z is ∞ $\sum_{t\in T}\prod_{r\in t}\dots$

CKY-Algorithm: Normalizer of parse ↳ No longer infinite trees because $N\to N\Rightarrow O(\infty^N)$

Runs in $O(N^3\cdot\#Rules)$ | Run in (max, x) semiring = score of best parse | $(\{0,1\}, \vee, \wedge, 0, 1)$ ← Boolean check string ∈ grammar

Dependency grammar: link word with its syntactic head | Dependency tree: directed spanning tree: no cycles, one input arc

Depend. vs. Constituency: CFGs not provides info about syntactic head, dep. tree no info about constituency structure

Projective depend. tree: no crossing arcs, closely related to constituency, DP algos for parsing like CKY

Non-projective: crossing arcs, closely related discontinuous constituents ← $O(n^n)$: Spanning Tree: n^{n-2} + Prod. dist. over non-projective: $p(t|w) = 1/z \cdot \exp(\text{score}(t,w))$ | $Z = \sum_{t' \in T(w)} \exp(\text{score}(t',w))$ $(n-1)^{n-2}$

Edge factored assumption: $p(t|w) = \frac{1}{z} \cdot \left[\prod_{(i \to j) \in T} \exp(\text{score}(i,j,w))\right] \cdot \exp(\text{score}(r,w))$ | $A_{ij} = \exp(\text{score}(i,j,w))$ | $p_j = \exp(\text{score}(j,w))$

$Z = \sum_{t' \in T(w)} \prod_{(i \to j) \in t'} \exp(\dots) \cdot \exp(\dots)$ ← compute — Score of tree — Score of root

Kirchhoff's Matrix Tree Theorem: # undirected spanning trees in undirected graph in $O(n^3)$ time: $L_{ij} = p_j$ if $i = 1$; $\sum_{i'=1, i' \neq j}^{n} A_{i'j}$ if $j = j' > 1$; $-A_{ij}$ otherwise; $Z = |L|$

Chu-Liu-Edmonds Algo: Kruskal's algo to directed trees in $O(n^3)$ | Kruskal in $O(E \cdot \log E)$

Semantic Parsing: meaning of sentence ⇒ Lambda calculus | Syntactic ⇒ grammer

Transliteration: maps string from one character set to another: replace character with approx. phonetic equivalent

Weighted-finite-state-transducers: $T = \langle Q, \Sigma, \Lambda, \lambda, \rho, \delta \rangle$ (Q: states, Σ: input vocab, Λ: output vocab, λ: state to initial score, ρ: state to final score, δ: transition)

ambiguous: more than one accepted path $\Pi(x \in \Sigma^*, y \in \Lambda^*)$ | WFST distr.: $p(y|x) = \frac{1}{Z} \cdot \exp(\text{score}(y,x)) \overset{*}{=} \frac{1}{Z} \cdot \sum_{\pi \in \Pi(x,y)} \exp(\sum_{n=1}^{|\pi|} \text{score}(\tau_n))$

*Structure assumption: score (π: seqn. of state transitions) $= \sum_{n=1}^{|\pi|} \text{score}(\tau_n)$ | $\Pi(x|y)$ = all sequences $x \to y$

Normalizing constant: $Z = \sum_{y \in \Lambda^*} \exp(\text{score}(y,x)) \overset{*}{=} \sum_{y' \in \Lambda^*} \sum_{\pi' \in \Pi(x,y')} \exp(\sum_{n=1}^{|\pi'|} \text{score}(\tau_n))$ | $y' \in \Lambda^*$ all words in machine | $\pi' \in \Pi(x,y')$ all paths $x \to y'$

Floyd-Warshall Algo: computes Λ^*: all pairs shortest paths + no negative cycles in $O(n^3)$

↳ Same as simple matrix multiplication with semiring $\langle \mathbb{R}^+ \cup \{+\infty\}, \min, +, +\infty, 0 \rangle$ with non-negative weights (tropical ring)

1. for $n: 1 \to N$ | 2. for $p: 1 \to N$ | 3. sum ← $+\infty$ | 4. for $m: 1 \to N$ | 5. sum = min(sum, W[n][m] + W[m][p]) | 6. $W^2[n][p]$ = sum

W^k encodes the shortest path between two nodes of length k | all-pair shortest paths: $\min(W^0, W^1, \dots, W^{\#nodes})$

Floyd-Warshall as matrix mult: 1. for each vertex k in N: $W^k[k][k] = 0$ | 2. for each vertex k in N: sum = min(sum, $W^{k-1}[n,m] + W^1[m,p]$)

Floyd-Warshall-Kleene: $W^0 \oplus W^1 \oplus \dots \oplus W^\infty = W^*$ (= Kleene closure) | $a^* = 1/(1-a)$ (= Kleene Operator) 3. $W^k[n,p]$ = sum

$W^* = \bigoplus_{k=0}^{\infty} W^k = I + W \bigoplus_{k=0}^{\infty} W^k = I + WW^* \Rightarrow W^* - WW^* = I \Rightarrow (I-W)W^* = I \Rightarrow W^* = (I-W)^{-1}$ (same as LU decomp.)

1. for each edge (u,v) do: dist[u,v] = W[u,v] $(= W^{-1})$ | 2. for each vertex v do: dist[v,v] = W[v,v]

3. for $k = 1 \to N$, for $i = 1 \to N$, for $j = 1 \to N$: $dist[i,j] = dist[i,j] \oplus (dist[i,k] \otimes dist[k,k] \otimes dist[k,j])$ $O(n^3)$

$Z = \sum_{y' \in \Lambda^*} \exp(\text{score}(y',x)) = \sum_{y' \in \Lambda^*} \sum_{\pi' \in \Pi(x,y')} \exp(\sum_{n=1}^{|\pi'|} \text{score}(\tau_n)) = \alpha^T (\sum_{\omega \in \Lambda \cup \{\varepsilon\}} W^\omega) \cdot \beta$

We optimize score weights with log-likelihood: $\sum_{i=1}^{I} \log(p(y^i, x^i)) = \sum_{i=1}^{I} \text{score}(y^i, x^i) - \log Z(x^i)$

Sequence-to-sequence Model: $z = \text{encoder}(x)$ | $y|x = \text{decoder}(z)$ | $p(y|x) = \prod_{t=1}^{T} p(y_t | x, y_1, \dots, y_{t-1})$

Train model params: $\text{argmax}_\theta \sum_{i=1}^{N} \log p(y^i | x^i; \theta) = \text{argmax}_\theta \sum_{i=1}^{N} \sum_{t=1}^{|y^i|} \log(p(y_t^i | x^i, y_{<t}^i, \theta)$ (=LSTM)

The last hidden state of encoder is first hidden state of decoder ⇒ $y_1 = p(\cdot|x)$ ⇒ $y_2 = p(\cdot, x, y_1)$ ⇒ ...

Decoder only receives a single vector from decoder ⇒ information bottleneck ⇒ Attention Model

Attention: keep hidden vector of decoder at each time step. Weights for each hidden vector in encoder at each time-step | Weights computed in query to key-values from training data

attention weights: $\alpha_i^{(t)} = \exp(\text{score}(q_t, k_i)) / \sum_j \exp(\text{score}(q_t, k_j))$ ⇔ $\alpha^t = \text{softmax}(\text{score}(q_{t-1}, K))$

context vector $c^t = \sum_j \alpha_i^{(t)} \cdot v_j$ ⇔ $c^t = \alpha^{(t)T} \cdot V$ | $k_i = v_i = h_i^1$ (= hidden state) | $q_t = h_t^1$ (= decod. at position t)

$K = V = H^e$ (= Horizontally stacked encoder representation) | Best translation $y^* = \text{argmax}_{y \in \Sigma^*} \text{score}(x,y)$

$\text{score}(x,y) = \log(p(y|x))$ | $1 \cdot \sum_{t=1}^{|y|} \log p(y_t | x, y_{<t})$ (=Non-markovian structure) | $p(y_t | x, \langle y_{t-m}, \dots, y_{t-1} \rangle$

e.g. language translation, image captioning, — ↳ $O(|\Sigma|^{n_{max}})$ → summarization ↳ $O(|\Sigma| \cdot n_{max})$ ⇒ DP

Probabilistic Model: make assumption about distr. of data and independence | Non-prob.: more interpretable no uncertainty

Discriminative: model decision boundary (CRF, RNN) | Generative approach: model dist. (n-gram, Markov Random Field)

Learned: Perceptron, SVM | Manually Crafted: context-free-grammer | Independence: allow DP → (global normalization), not needed for NN

Locally normalized: Ngram, RNN: $1/z \cdot \text{score}(x_i, y_i)$ | Globally: CRF, MRF: $1/z \cdot \text{score}(x,y)$ | not very realistic for some

$L_1 = L(\theta) + \lambda \|\theta\|_1$: coeffs to be 0 | $L_2 = L(\theta) + \lambda \|\theta\|_2^2$: small non-zero | Extrinsic: does model improve final task? | Problems

Intrinsic NLP evaluation: based on criteria, more general e.g. log-likelihood

Cost NN: $O((C(N+1)) \cdot K_1 + \sum_{L=1}^{L-1} (K_L + 1) \cdot K_{L+1}) + (K_L + 1) \cdot C)$ | Prior: $p(\theta|D) = \frac{p(\theta) \cdot p(D|\theta)}{p(D)} \propto p(\theta) \cdot p(D|\theta)$

Classification functions: $x_1 - x_2 \leq 0$, $|x_1| + |x_2| < 2$, $x_1^2 + x_2^2 < 4$ | Sigmoid $\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$

$\tanh'(x) = 1 - \tanh^2(x)$ | $\text{ReLu}'(x) = 1$ if $x > 0$ else 0 | ≤ 0 | XOR: $y = 1 \cdot \text{ReLu}(1 \cdot x_1 + 1 \cdot x_2) - 2 \cdot \text{ReLu}(x_1 + x_2 - 1)$

$p(\text{here are, you}) = p(\text{you are here}) / p(\text{you are})$ | $2^{-1} = 0{,}5$ | $2^{-2} = 0{,}25$ | $2^{-5} \cdot 2^{-3} \cdot 2^{-1} = 2^{-9}$ ⇒ log, semiring: $+^{\log} = \log(2^a + 2^b)$ (when summing) | $x = +\infty$ ← when mult

Semiring Check: $\langle A, +, 0 \rangle$ commutative monoid + $\langle A, \times, 1 \rangle$ monoid + 0 is annihilator for x

Viterbi: $v_n(t_n) = \max_{t_{n-1}} (\text{score}(\langle t_{n-1}, t_n \rangle, w) + v_{n-1}(t_{n-1}))$ $O(T^2 N)$: $V_2(N) = \max(V_1(N) + \text{score}(\dots), V_1(V) + \text{score}(\dots))$

α-conversion: rename variables: $\lambda x.(x\ y) \overset{\alpha}{\to} \lambda y.(y\ y)$ | β-reduction: avoid variables getting bound: outside → inside

Termination issue: $F = \lambda x.((x\ x)\ x)$: $(F\ P) = (\lambda x.((x\ x)\ x)\ F \to \dots$ | n-gram states: $m + |V|^{n-1} + 1$ | m = start state

Trigram Transition function: $\delta(q_{ij}, w, q_{ki}) = \log p(w_m = k | w_{m-1} = i, w_{m-2} = j)$ | $m = |V|^{n-2}$ for $n \geq 2$, 0 $n = 1$

Initial weight function: $\lambda(q_{ij}) = \log p(w_m = i | w_{m-1} = D, w_{m-2} = \square)$ | $\text{softmax} \frac{\exp(\theta \cdot f(x,y_0))}{\exp(\theta \cdot f(x,y_0)) + \exp(\theta \cdot f(x,y_1))} = \frac{\exp(\theta \cdot (f(x,y_0) - f(x,y_1)))}{1 + \exp(\theta \cdot (f(x,y_0) - f(x,y_1)))}$

RNN: $h^t = \text{softmax}(W_{hh} \cdot h^{t-1} + W_{xh} \cdot x^t + b)$ | $\hat{y}^t = \text{softmax}(W_{hy} \cdot h^t + b)$ | Loss: $L = \sum L^t = \sum_{t=1}^{T} L(y^t, \hat{y}^t)$

$\frac{\partial L^t}{\partial W} = \sum_{k=1}^{t} \frac{\partial L^t}{\partial y^t} \frac{\partial y^t}{\partial h^t} \frac{\partial h^t}{\partial h^k} \frac{\partial h^k}{\partial W}$ | $\frac{\partial h^t}{\partial h^k} = \prod_{i=k+1}^{t} \frac{\partial h^i}{\partial h^{i-1}}$ | Attention: $O(dNM)$

Encoder-Decoder: $h_m^{(s)} = f(W_1^s \cdot h_{m-1}^s + W_2^s \cdot x_m) \Rightarrow h_0^t = h_N^s \Rightarrow h_m^t = f(W_1^t \cdot h_{m-1}^t + W_2^t \cdot y_{m-1})$ | $O(d^2 \cdot \text{Layers} \cdot \text{Input-length}_s)$

$\frac{\partial \text{Loss}}{\partial h_m^{(s)}} = \frac{\partial \text{Loss}_{m+1}}{\partial h_m^t} \cdot \frac{\partial h_m^t}{\partial h_{m-1}^t} \cdot \dots \cdot \frac{\partial h_2^t}{\partial h_1^t} \cdot \frac{\partial h_1^t}{\partial h_N^s} \cdot \dots \cdot \frac{\partial h_2^s}{\partial h_{N-1}^s} \cdot \dots \cdot \frac{\partial h_{m+1}^s}{\partial h_m^s}$ | $\frac{\partial h_m^t}{\partial h_{m-1}^t} = \frac{\partial f}{\partial h_{m-1}^t} \cdot (W_1^t \cdot h_{m-1}^t + \dots) \cdot W_1^t$

CRF vs. MEMM = locally vs. globally normalized | Jacobian: $J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$

Exponential family: gaussian, Poisson, categorical | $x^{\alpha-1} = \exp(\log(x) \cdot (\alpha - 1))$