1 Probability Calculus

$p(x|y) = \frac{p(x,y)}{p(y)} = \frac{p(x)p(y|x)}{\sum_x p(x)P(y|x)} = \frac{Prior \times Likelihood}{Evidence}$

$p(x) = \sum_z p(x,z) \qquad p(x,y,z) = p(x)p(y|x)p(z|x,y)$

$x \perp\!\!\!\perp y \ \Rightarrow\ p(x,y|z) = p(x|z)p(y|z) \ \wedge\ p(x|y,z) = p(x|z)$

$E[kf(x) + ng(x)] = kE[f(x)] \cdot nE[g(x)]$

$Cov(X) = E(X^2) - E(X)^2 \qquad Cov(aX+b) = a^2 Cov(X)$

$E_{Y|X=x_i}[f(x_i,Y)] = \sum_{y' \in Y} f(x_i,y')p(y'|x_i,\theta)$

2 Semirings $R = (A, \oplus, \otimes, \bar{0}, \bar{1})$

2.1 $(A, \oplus, \bar{0})$ is a commutative monoid

$(a \oplus b) \oplus c = a \oplus (b \oplus c)$ $\bar{0} \oplus a = a \oplus \bar{0} = a$ $a \oplus b = b \oplus a$

2.2 $(A, \otimes, \bar{1})$ is a monoid

$(a \otimes b) \otimes c = a \otimes (b \otimes c) \qquad \bar{1} \otimes a = a \otimes \bar{1} = a$

2.3 $\otimes$ distributes over $\oplus$: for all a, b, c in A

$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$

$c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$

2.4 $\bar{0} \otimes a = a \otimes \bar{0} = \bar{0}$

Semiring	Set	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$	intuition/application
Boolean	$\{0,1\}$	$\vee$	$\wedge$	0	1	logical deduction, recognition
Viterbi	$[0,1]$	max	$\times$	0	1	prob. of the best derivation
Inside	$\mathbb{R}^+ \cup \{+\infty\}$	$+$	$\times$	0	1	prob. of a string
Real	$\mathbb{R} \cup \{+\infty\}$	min	$+$	$+\infty$	0	shortest-distance
Tropical	$\mathbb{R}^+ \cup \{+\infty\}$	min	$+$	$+\infty$	0	with non-negative weights
Counting	$\mathbb{N}$	$+$	$\times$	0	1	number of paths

Real for log-probabilities: $\langle \mathbb{R} \cup \{-\infty\}, max, +, -\infty, 0\rangle$

Real for max-probabilities CKY: $\langle \mathbb{R}^+, max, \times, 0, 1\rangle$

Real for LogSumExp: $\langle \mathbb{R} \cup -\infty, log_+, +, -\infty, 0\rangle$

3 Backpropagation (Chainrule, DP)

$\frac{\partial f}{\partial g} = \frac{\partial f}{\partial h} \cdot \frac{\partial h}{\partial g} \qquad \frac{\partial f_k}{\partial g_i} = \sum_{j=1}^m \frac{\partial f_k}{\partial h_j} \cdot \frac{\partial h_j}{\partial g_i}$

Constr. Th.: Same asympt. complexity as the orig. func.

4 Log-Linear Modeling (Softmax'd dotproduct)

$p(y|x) = \frac{count(x,y)}{count(x)} \qquad p(y|x,\theta) = \frac{exp(\theta \cdot f(x,y))}{\sum_{y' \in Y} exp(\theta \cdot f(x,y'))}$

Binary: $p(y|x,\theta) = \sigma(\theta \cdot x) = 1/(1 + exp(-\theta \cdot x))$

$\theta_{MLE} = \underset{\theta \in \Theta}{argmax}\, L(\theta) \qquad L(\theta) = \prod_{n=1}^N p(y_n|x_n,\theta)$

$LL(\theta) = \sum_{n=1}^N log\ p(y_n|x_n,\theta) \qquad NLL(\theta) = -LL(\theta)$

$\frac{\partial NLL(\theta)}{\partial \theta_k} = -\underbrace{\sum_{n=1}^N f_k(x_n,y_n)}_{observed} + \sum_{n=1}^N \underbrace{\sum_{y' \in Y} p(y'|x_n,\theta) f_k(x_n,y')}_{expected}$

Expectation matching: observed = expected

Hessian: $\nabla^2 NLL(\theta) = \sum_{i=1}^n Cov(f(x_i,Y))$

Softmax$(h,y,T) = \frac{exp(h_y/T)}{\sum_{y' \in Y} exp(h_{y'}/T)}, \ \ h_y = \theta \cdot f(x,y)$

$T \to \infty$: uniform, max entropy

$T \to 0$: annealing, max function, minimum entropy

Softmax is diffbar for $T > 0$

Exp. family: $p(x|\theta) = \frac{1}{Z(\theta)} h(x) exp(\theta \cdot \phi(x))$

finite sufficient stat., conjugate priors, max. entropy distr.

Skip-Gram: $p(c|w) = \frac{1}{Z(c)} exp(e_{wrd}(w) \cdot e_{ctx}(c))$

4.1 Hessian Matrix

Jacobian $\nabla f(x)$: $J_{ij} = \frac{\partial f_i}{\partial x_i}$ **Hessian** $\nabla^2 f(x)$: $(H_f)_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$

Trick: $e_i^T \nabla^2 f(x) = \nabla(e_i^T \nabla f(x))$

Recursion: $O(mn^{k-1})$

5 Feed-forward NN: $\sigma(W_i^T \cdot ReLU(W_j^T x + b_j) + b_i)$

Non-linearity + learning the structure, feature engineering is tedious because we don't know the structure, thus feature extractor (architecture) engineering

Will not learn if weights are all initialized 0 or the same.

Finite-difference procedure:

$O((((n+1) \cdot k_1 + (\sum_{l=1}^{L-1}(k_l+1) \cdot k_{l+1}) + (k_L+1) \cdot c)^2)$

5.1 Activation Function (non-linearities)

$\sigma(x) = 1/(1+exp(-x)) \qquad \sigma'(x) = \sigma(x)(1-\sigma(x))$

$tanh(x) = 2\sigma(2x) - 1 \qquad tanh'(x) = 1 - tanh^2(x) = 1/(x^2+1)$

$ReLU(x) = max(0,x)$, PReLU, ELU, SELU, SoftPlus

Dead neurons: ReLU with $x < 0 \to$ PReLU, Leaky ReLU

5.2 Vanishing Gradient

High (absolute) input values (sigmoid, tanh, RNN) $\to$ partial derivatives $\sim 0 \to$ abs(weights) $\ll 1 \to$ model stops learning eventually.

Solution: ReLU, LSRM, GRU, fewer layers, batch norm

5.3 Exploding Gradient

Large increase in the norm of the gradient during training (NN, RNN). All gradients $> 1 \to$ large updates $\to$ overflow.

Solution: fewer layers, regularization

6 RNN

$h_t = f(x_t, h_{t-1})$, with cell f (RNN, GRU, LSTM)

$\frac{\partial h_{m+k}}{\partial h_m} = \prod_{i=0}^{k-1} \frac{\partial h_{m-i}}{\partial h_{m-i-1}}$

RNN: $h_t = tanh(\theta h_{t-1} + x_t)$ (or $\sigma()$)

GRU tries to solve the vanishing/exploding gradient problem.

7 Language Modeling

Preprocessing: Tokenization, Lower casing, Stemming, Stop word removal, reducing vocabuulary

Features: n-grams, one-hot-encoding, bag-of-words, word embedding, bag of embeddings

Locally normalized: $Z = \sum_{y' \in V^*} \prod_{t=1}^{|y'|} \theta_{y'_{\leq t}} = 1$

Globally normalized: Floyd-Warshall-Kleene, MEMM

7.1 n-gram

$p(y_t|y_{<t}) = p(y_t|y_{t-1},\dots,y_{t-n+1}) = \text{softmax}(h_{y_t})$

Trade-off: small $n \to$ high bias big $n \to$ high variance

States: $m + |V|^{n-1} + 1, m = |V|^{n-2, n>1}$ or 0 $\qquad O(|V|^{n-1})$

Bengio et al. 2003: Language model as neural network, use e word embeddings in MLP, using neural parametrization of an n-gram model

$h = b_1 + W_1 \cdot e(hist) + W_2 \cdot tanh(b_2 + W_3 \cdot e(hist))$

8 Conditional Random Fields (CRFs) (locally normalized)

Part-of-speech (POS) tagging: Adj., Nouns, Verbs, etc.

Use graph for score$(< D, N, V, N >, w)$

$p(t|w) = \frac{exp(score(t,w))}{\sum_{t' \in T} exp(score(t',w))}$

score$(t,w) = \sigma \cdot f(t,w)$ or $= NN_\sigma(t,w)$

Structure assumption: $p(t|w) = \frac{exp(\sum_{n=1}^N score(\langle t_{n-1}, t_n\rangle, w))}{\sum_{t' \in T^N} exp(\sum_{n=1}^N score(\langle t'_{n-1}, t'_n\rangle, w))}$

$LL(\theta) = \sum_{i=1}^I score(t^{(i)}, w^{(i)})) - T\ log \sum_{t' \in T^N} exp \frac{score(t', w^{(i)})}{T}$

For $T \to 0$: Viterbi (structured perceptron)

8.1 Computing the normalizer (DP)

$\beta(w, t_N) \leftarrow 1$

for $n \leftarrow N-1, \dots, 0:$

$\quad \beta(w, t_n) \leftarrow \oplus\ exp(score(\langle t_n, t_{n+1}\rangle, w)) \otimes \beta(w, t_{n+1})$

Complexity: $O(|tagset|^{'n \text{ of n-grams}'} |sentence|)$

9 Constituency Parsing

9.1 Context-free grammar: $G = \langle N, S, \Sigma, R\rangle$

Rules can be applied to a non-terminal regardless of context.

N – set of nonterminal symbols

S – start symbol

Σ – set of terminal symbols

R – set of production rules

Problem: I like to play bridge and bob chess $\to$ cross-serial dependency does not work

9.2 Chomsky normal form: $(N_1 \to N_2 N_3)$ and $(N \to a)$

9.3 Probabilistic CFG

Sum over a rule = 1, locally normalized, probability will be multiplied over tree

9.4 Weighted CFG

Non-negative, globally normalized, weight will be $exp()$ multiplied over tree, structured softmax

$p(t)$ is infinite, $p(t|s)$ finite if no cycles (CNF)!

$p(t|s) = \frac{\prod_{r \in t} exp(score(r))}{\sum_{t' \in T(s)} \prod_{r' \in t'} exp(score(r'))}$

9.5 CKY $O(N^3|R|)$

SemiringCKY$(\mathbf{s}, \langle \mathcal{N}, S, \Sigma, \mathcal{R} \rangle, \text{score})$:

```
N ← |s|
chart ← 0
for n = 1, ..., N :
    for X → s_n ∈ 𝒩 :
        chart[n, n+1, X] ⊕= exp{score(X → s_n)}
for span = 2, ..., N :
    for i = 1, ..., N − span :
        k ← i + span
        for j = i+1, ..., k−1 :
            for X → Y Z ∈ 𝒩 :
                chart[i, k, X] ⊕= exp{score(X → Y Z)} ⊗ chart[i, j, Y] ⊗ chart[j, k, Z]
return chart[0, N, S]
```

10 Dependency Parsing

Dependency Tree = Directed Spanning Tree
All non-root nodes have one incoming edge, no cycles, only one outgoing from root
Projective dependency trees: no crossings, close to constituency, use cky
Non-projective dependency trees: crossings, close to discontinued constituents, n^{n-2} Spanning trees, $(n-1)^{n-2}$ root constraints $\rightarrow O(n^n)$

Matrix-Tree Theorem: $L_{ij} = \overset{i \neq j}{-A_{ij}} \mid \overset{i=j}{(\rho_j +) \sum_{k \neq i} A_{kj}} \mid \overset{i=1}{\rho_j}$
$p(t|w) = \frac{\prod_{(i \rightarrow j) \in t} exp(score(i,j,w))}{|L|} \rightarrow O(n^3)$

10.1 Chu-Liu Edmonds Algoritm $O(n^2)$

1. Greedy selection (best incoming for each node, except root)
2. On cycle $\rightarrow$ contract (cycle through mult. incoming, sum)
3. For mult. root, find replacement with lowest swap cost

11 Lambda Calculus (Free and Bound)

Appl.: $M\,N\,P \equiv ((M\,N)\,P)$ **Abstr.**: $\lambda x.M\,N \equiv \lambda x.(M\,N)$
$A \rightarrow B \equiv \neg A \vee B \equiv \neg(A \wedge \neg B)$ **α-conversion**: $\lambda x.x \rightarrow \lambda y.y$
β-reduction: $\lambda x.x\,z \rightarrow z$ **I-combinator**: $\forall x: (I\,x) = x$
K-combinator: $\forall x,y: (K\,x\,y) = ((K\,x)y) = x$ $(S\,K\,K) \equiv I$
S-combinator: $\forall x,y,z: (S\,x\,y\,z) = (x\,y(y\,z)) = ((x\,z)(y\,z))$
B-combinator: $\forall x,y,z: (B\,x\,y\,z) = (x(y\,z))$
C-combinator: $\forall x,y,z: (C\,x\,y\,z) = ((x\,z)y)$
T-combinator: $\forall x,y: (T\,x\,y) = (y\,x)$

12 Weighted Finite-State Transducers (WFST)

Transliteration: map to another character set (alphabet)
$T = \langle Q, \Sigma, \Omega, \lambda, \rho, \delta \rangle$
Q: all states (finite set)
Σ: input vocabulary
Ω: output vocabulary
λ: function map to init scores
ρ: function map to final scores
δ: transition function, mapping transitions (arcs) to scores
$Z = \alpha^T \left(\sum_{\omega \in \Omega \cup \varepsilon} W^{(\omega)} \right)^* \beta$
$\sum_{i=1}^{I} log\ p(y^{(i)}|x^{(i)}) = \sum_{i=1}^{I} score(y^{(i)}, x^{(i)}) - log\ Z(x^{(i)})$

12.1 Floyd-Warshall-Kleene Algorithm $O(n^3)$

```
let dist be a N × N array of minimum distances initialized to 0 (infinity)
for each edge (u, v) do
    dist[u][v] ← W[u][v]  // This corresponds to W^1
for each vertex v do
    dist[v][v] ← W[v][v] // This corresponds to W^0
for k from 1 to N
    for i from 1 to N
        for j from 1 to N
            dist[i][j] ←dist[i][j] ⊕ (dist[i][k] ⊗ dist[k][k]* ⊗ dist[k][j])
```

13 Sequence-to-Sequence Models

$z = encoder(x) \qquad y \mid x \sim decoder(z)$
Enc: $\underset{\theta}{argmax} \sum_{i=1}^{N} log\ p(y^{(i)}|x^{(i)}, \theta)$
$= \underset{\theta}{argmax} \sum_{i=1}^{N} \sum_{t=1}^{|y^{(i)}|} log\ p(y_t^{(i)}|x^{(i)}, y_{<t}^{(i)}, \theta)$
Dec: $score := log\ p(y|x) = \sum_{t=1}^{|y|} log\ p(y_t|x, \langle y_{t-n}, \ldots, y_{t-1} \rangle)$
For one y: $O(|\Sigma| \cdot n_{max})$, for all y: $O(|\Sigma|^{n_{max}})$
$\rightarrow$ beam search, sampling, greedy search
Only last hidden state of seq. is passed to decoder!
Used for: mach. transl., text summarization, img captioning

13.1 Attention Mechanism

RNN+NN: RNN gets h_{t-1} & previous NN output, NN gets h_t and attention vector (all h_i of enc, weighted with softmax, done for each dec step)
Weights: $\alpha^{(t)} = softmax(score(q_t, K))$
Context: $c^{(t)} = \alpha^{(t)T} V$
Vec rep (hid. state) produced by enc at pos i: $k_i = v_i = h_i^{(e)}$
Vec rep produced by dec at pos t: $q_t = h_t^{(d)}$
= stacked encoder vector representations: $K = V = H^{(e)}$

14 Axes of Modeling

14.1 Probabilistic models

Prob. distr. over classes (outcomes)
+ probability theory, convenient & intuitive framework
- assump. about distr. (indep./distr. of noise) may be false
Discriminative: models decision boundary (CRF, RNN)
Generative: models distr. of class (n-gram, MRF, RNN)
Structured predictors: Use if decomposing output space is helpful (output space is too big)
Locally Normalized: + efficient to train, only prediction of current state, - label bias (n-gram, RNN)
Globally Normalized: + scores at each time step can have diff. importance, - comp. global norm. constant structural indep. assump. are crit. (CRF, MRF, RNN)

14.2 Non-probabilistic models

Separate feature space and return the class associated with the space where they believe a sample comes from
+ more interpretable
- no direct way to quantify uncertainty of a prediction
Learned: Perceptron, SVM
Manually-crafted: CFG, LIG

14.3 Bias/Variance Trade-off

A biased simpler model can have better overall performance than a complicated unbiased model (due to lower variance of the simpler model)

14.4 Loss-Function

Convexity, continuity, diffbar, comp. complexity, sensitive to noise and outliers, connection to end goal, tradeoff between classes
cross-entropy loss func. $\equiv$ *negative log-likelihood of model*
MLE: efficient to evaluate, consistent, asympt. efficient, only for probabilistic models, can easily overfit (bad on unseen)
Alternatives: Maximum margin (hinge), logistic, exponential

14.5 Regularization (improve Generalization)

Adding prior information to prevent overfitting to noise
Deep Learning: Weight Decay, Dropout, Early Stopping, Batch normalization
Lasso L1: $L(\theta) + \lambda||\theta||_1$, encourages many coeff to exactly zero, not diffbar
Ridge L2: $L(\theta) + \lambda||\theta||_2^2$, shrinks params to small non-zero values
Interpretation $\theta \sim \mathcal{N}(0, \tau^2 I)$: τ^2 is a measure of confidence in our prior; the lower τ^2, the more "belief" we place in our prior relative to the data. Since in the case of ridge, our prior mean is zero, this means that the lower we choose τ^2 the stronger we bias our resulting coefficients toward the prior mean (i.e., zero). Thus, since σ^2 is fixed, the lower we choose τ^2 the stronger the regularization and vice versa (i.e., high τ^2 implies little confidence in our prior and thus low regularization).