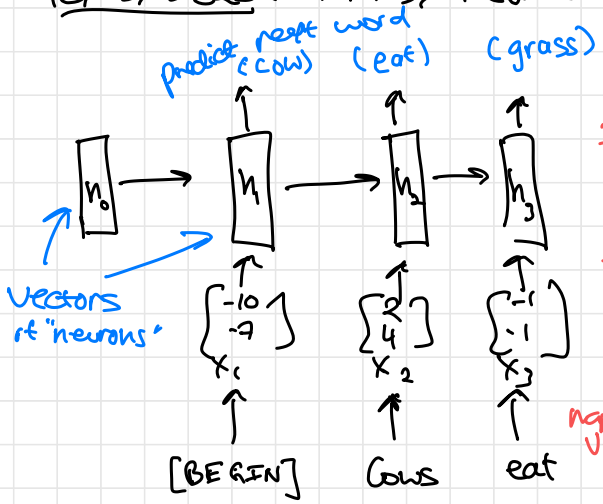
10/5/2023: RNNs, Review for Midterm



multiclass classification ≈ softmax regression

RNN recursion

$h_t = \tanh(W^h h_{t-1} + W^x x_t + b)$

map each word to vector

learned by gradient descent

Model Params

weight vector for each class

classes = # vocab words

W^h, W^x matrix

h_0, b vector

word vector parameters

"Dogs" → [2, 3]

"Cows" → [2, 4]

"eat" → [-1, -1]

"[BEGIN]" → [-10, -7]

Review

All cases: Training data $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$

Map $x \to y$

	Regression	Binary Classification	Multi-class Classification
What is y?	$y \in \mathbb{R}$	$y \in \{-1, 1\}$	$y \in \{1, 2, 3, \ldots, C\}$ eg. Generating text 1 word at a time
Corresponding linear model	Linear regression	Logistic Regression	Softmax regression
Parameters	$w \in \mathbb{R}^d$ dimension of x	$w \in \mathbb{R}^d$	$w^{(1)}, \ldots, w^{(C)} \in \mathbb{R}^d$ $C \times d$ total params
Probabilistic Story	$y \sim \text{Normal}(w^T x, \sigma^2)$ mean	$p(y=1 \mid x) = \sigma(w^T x)$ 	$p(y=j \mid x) = \frac{\exp(w^{(j)T} x)}{\sum_{k=1}^{C} \exp(w^{(k)T} x)}$ Normalize to a probability distribution
Loss Function measures how bad a choice of parameters is	Maximum likelihood Estimation (MLE) maximize probability of data w.r.t parameters θ: $\prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta)$ all params of model $\Leftrightarrow$ minimizing negative loglikelihood: $-\sum_{i=1}^{n} \log p(y^{(i)} \mid x^{(i)}; \theta)$		
How to minimize loss/ choose parameters	Gradient Descent OR Normal Equations	Gradient Descent	

Modeling Strategies

	Linear	Kernelized	Neural
Predictions	$w^T x$ (param, input) or $w^T \phi(x)$ (add more features)	$\sum_{i=1}^{n} \alpha_i k(x^{(i)}, x)$ (α_i: param; $x^{(i)}$: training example; x: input). Base prediction on similarity between x & $x^{(i)}$'s	$w^T f(x)$ (param; learned non-linear function of x)
How to make complex prediction?	Add features manually "feature engineering"	Choose polynomial or RBF kernel, instead of dot product	Add layers + non-linearities. Train layers to learn new features. Only one where we learn features from data ⇒ needs most data
Efficiency	Linear in size of $\phi(x)$ & #data	Kernel trick! but quadratic in #data	Linear in #data & #layers. SGD to take more steps on 1 loop over data
Bias/Variance	High bias: many functions are not linear	Low bias (universal approximators). High variance: easy to overfit	
Hyper-parameters. Not-learned, Chosen from dev set	Choice of features, L_1/L_2 regularization	Choice of kernel. L_2 reg.	#layers, #neurons/layer, which non-linearity: sigmoid/tanh/relu, Dropout, CNN: kernel size, RNN: word vector size