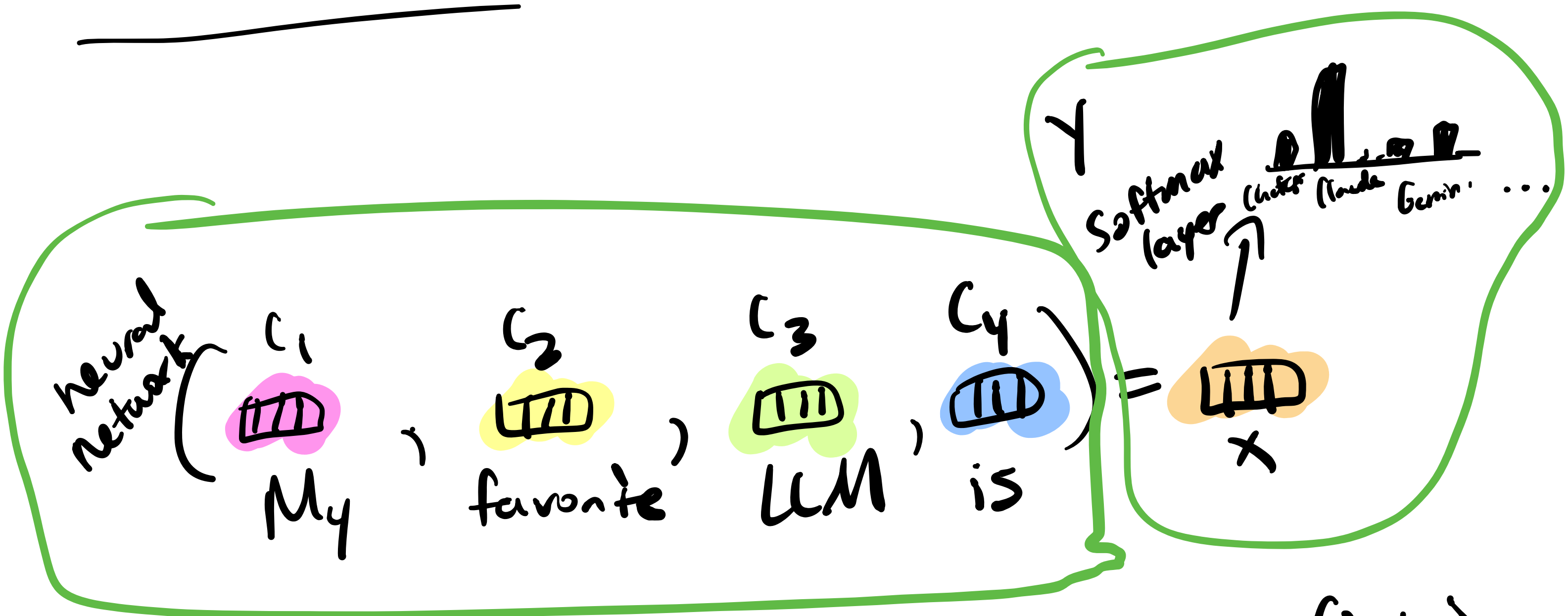
Logistics

- ↳ HW0, final project group assignments due Wed
- ↳ Exam 1: 10/20
  - Exam 2: 12/10
  - Final project report: 12/20



Composition

↓

1. element-wise functions
2. fixed-window neural LMs
3. recurrent neural LMs
4. Transformers
   - ↳ self-attention

$$y = \text{softmax}(Wx)$$

- y: V-dim
- softmax: takes an input vector and outputs a vector that is positive and sums to 1
- W: V×d
- x: d-dim

Composition function:

↳ input: sequence of d-dim embeddings corresponding to tokens of the prefix

↳ output: <u>single</u> vector representing the entire prefix

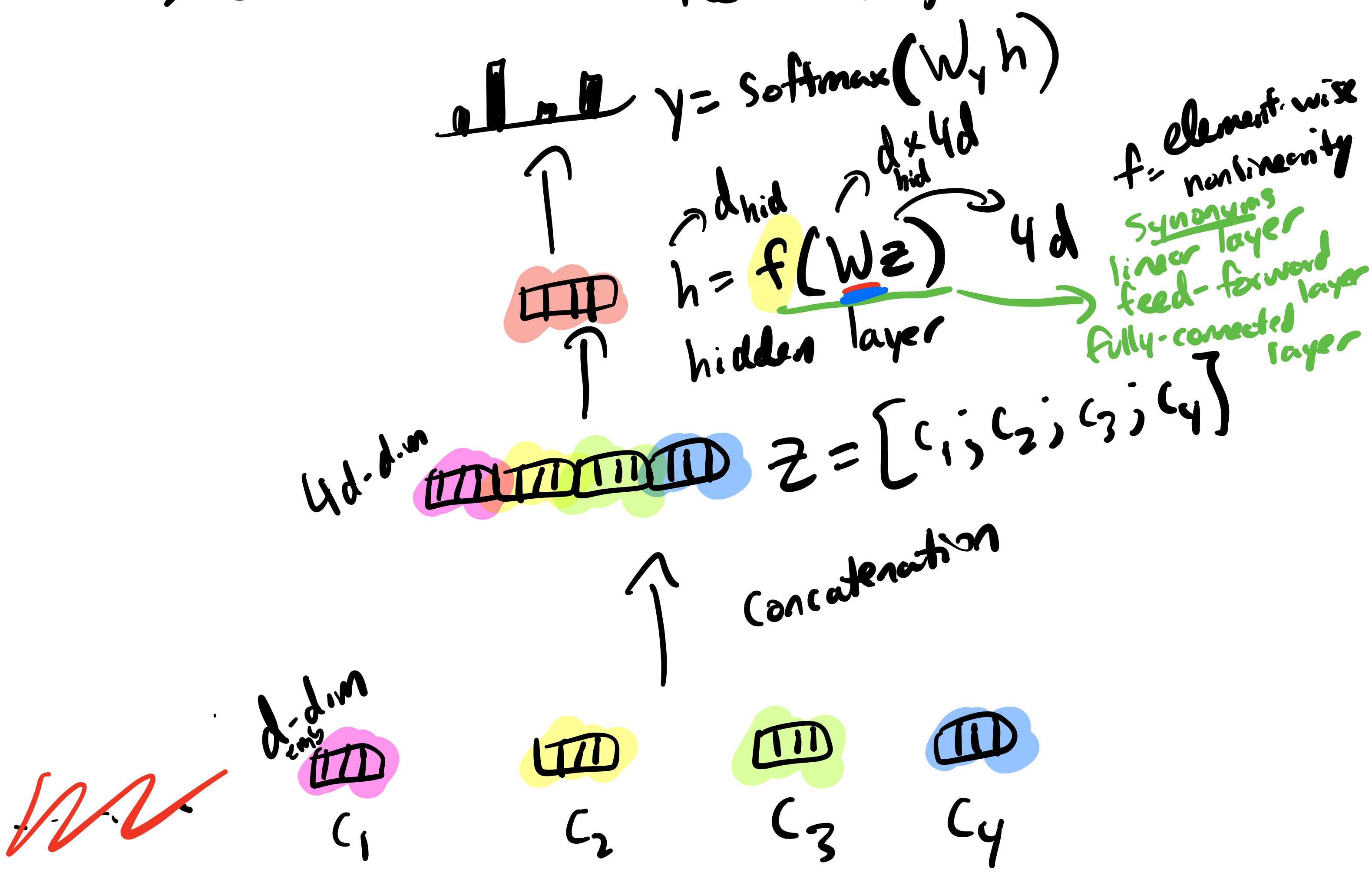
Simplest fn:

↳ element-wise sum.

$$x = \sum_{i}^{n} c_i$$

problems?

↳ concatenation: fixed-window neural LM
neural n-gram model



why nonlinearity?

↳ in LM (and many other tasks), the relationship between the input and the output is highly complex and nonlinear

↳ without nonlinearity, you can only model linear relationships

$$b = Wa$$
$$c = W_2 b$$
$$d = W_3 c$$
$$\vdots$$

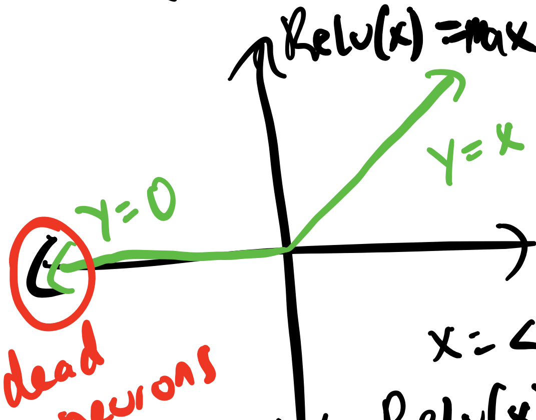
$$\Big\} \; c = \underbrace{W_2 W}_{} a$$

↳ $W_{new} a$

↑ simplification doesn't work if

$$b = f(Wa)$$
$$c = f(W_2 b)$$

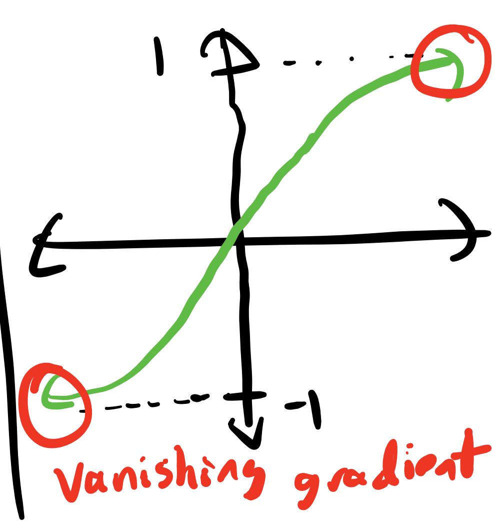
ReLU (rectified linear unit)



$x = <-2.3, 5.7, -50>$

$\text{ReLu}(x) = <0, 5.7, 0>$

tanh(x)

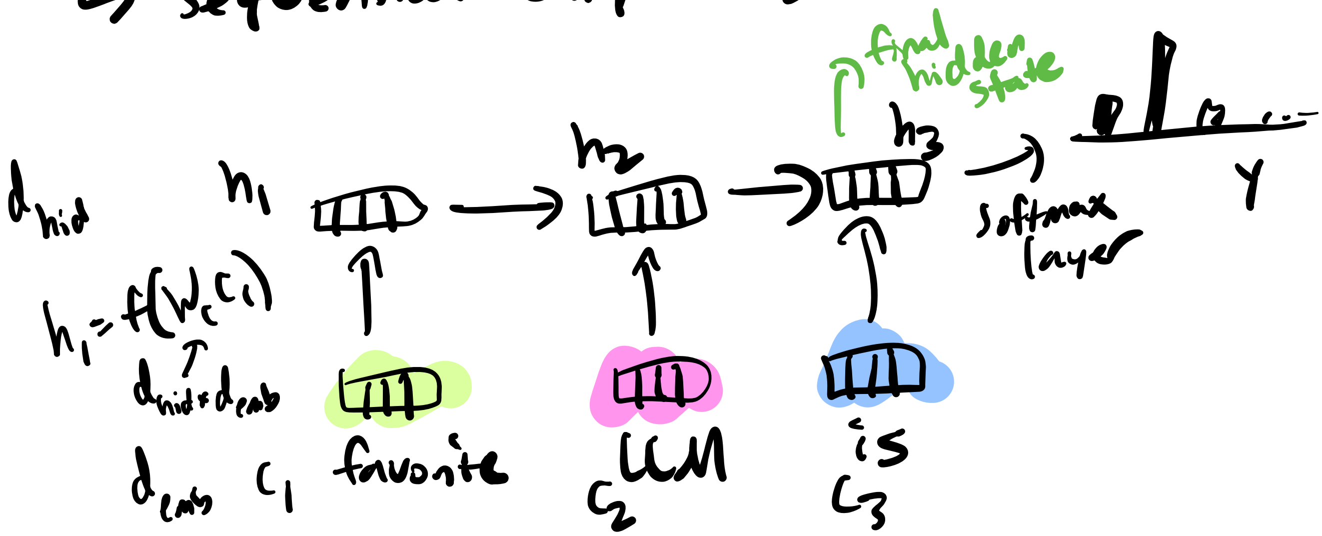


Comparing fixed-window NLM to n-gram model:

- storage: model size scales linearly with n rather than exponentially
- reduced sparsity problem
- hard to interpret
- fixed prefix window
  - impossible to handle long-range deps
- doesn't share weights between different positions in the prefix
  - 3 "My favorite LLM is" vs. "favorite LLM really is" 2

recurrent neural networks:

- sequential composition, one word at a time



↳ h_t ⇒ hidden state at timestep t
position

↳ h_t is a fn of h_{t-1} and c_t

h_{t-1} ↑ prev. hidden state

c_t ↑ current token embedding

↳ h_t and c_t don't have to be the same dimensionality

$$h_t = f(W_h h_{t-1} + W_c c_t)$$

f ← tanh

W_h ↑ $d_{hid} \times d_{hid}$

W_c ↳ $d_{hid} \times d_{emb}$

---

high-level training overview:

↳ NLMs contain trainable parameters

$$\theta_{rnn} = \{W_h, W_c, c_{1 \ldots v}, W_y\}$$

↓

1. randomly initialized
2. trained to better predict next word on a training dataset

- define a loss fn
  - tells us how bad we are doing at predicting the next word
  - $L(\theta) = -\log p(w_n | w_{1 \dots n-1})$
    - neg log prob of correct next word in training dataset
- given $L(\theta)$, we compute the gradient of L WRT θ
  - gradient provides direction of steepest ascent
- take a step in direction of negative gradient

$$W_{h_{new}} = W_{h_{OLD}} - \eta \frac{dL}{dW_h}$$

  - learning rate