DSCI 565: MODERN RECURRENT NEURAL NETWORKS; ATTENTION MECHANISMS AND TRANSFORMERS

Ke-Thia Yao

Lecture 13: 2025 October 8

Evaluation of Predicted Sequence: BLEU

translation metrics

- Bilingual Evaluation Understudy (BLEU) evaluates the *precision* of the predicted sequence against target sequence
- BLEU measures precision by checking what parts of the predicted sequences are actually in the target sequence
- Highest BLEU score is 1, exact match. Lowest score is 0.
- BLEU works by finding matches of n-grams in the predicted sequence and n-grams in the target sequence
- BLUE weights longer subsequence (larger n) matches more than shorter subsequence matches

Precision1-gram Example

- Target sequence: A, B, C, D, E, F
- Predicted sequences: A, B, B, C, D
- The precision for 1-gram $p_1 = \frac{4}{5}$ ← length of predicted sequence (n-gram)

Precision 2-gram Example

- Target sequence 2-gram: AB, BC, CD, DE, EF
- Predicted sequences 2-gram: AB, BB, BC, CD
- The precision for 2-gram $p_2 = \frac{3}{4}$

Precision 3-gram Example

- Target sequence 3-gram: ABC, BCD, CDE, DEF
- Predicted sequences 3-gram: ABB, BBC, BCD
- The precision for 3-gram $p_3 = \frac{1}{3}$

BLEU Score

- Let k be the longest n-gram for matching, BLEU is defined as

$$\exp\left(\min\left(0, 1 - \frac{\text{len}_{\text{label}}}{\text{len}_{\text{pred}}}\right)\right) \prod_{n=1}^{k} p_n^{1/2^n}$$

- Where $\text{len}_{\text{label}}, \text{len}_{\text{pred}}$ are the length of target and predicted sequence, respectively
- Example:

longer n-grams highly affect

$$\text{BLEU} = e^{\min\left(0, 1-\frac{6}{5}\right)} \left(\frac{4}{5}\right)^{\frac{1}{2}} \left(\frac{3}{4}\right)^{\frac{1}{4}} \left(\frac{1}{3}\right)^{\frac{1}{8}}$$

$$= e^{-1.2} \times .8944 \times .9306 \times .8717 = .8187 \times .7256 = 0.5941$$

BLEU Score

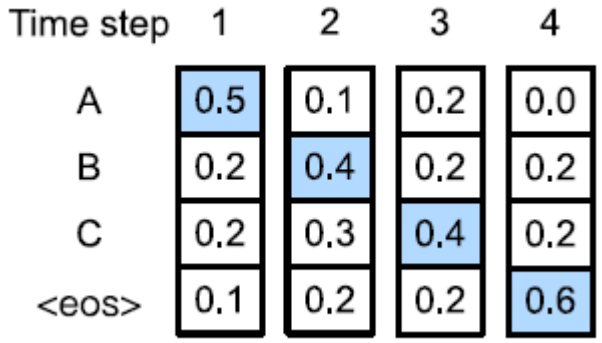
- BLEU is a popular metric for machine translation, and it is straight forward to compute
- But it does have disadvantages
  - Does not understand the meaning of the translation
  - Does not consider sentence structure
  - Requires exact matches (no synonyms, no stemming)

Search Strategies: Greedy Search

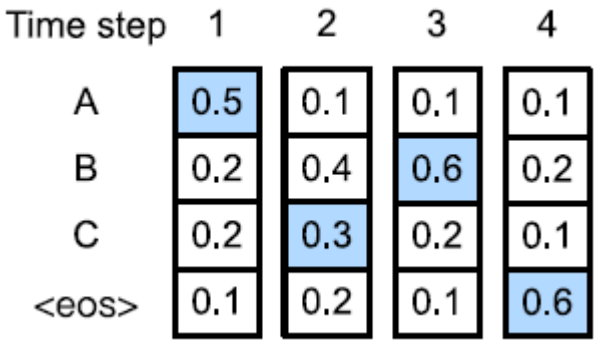
- Our implementation of `predict()` is based on greedy search
- At any timestep t', find the most likely token from vocabulary $\mathcal{Y}$

$$y_{t'} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} P(y | y_1, \ldots, y_{t'-1}, \mathbf{c})$$

Greedy: 0.5*0.4*0.4*0.6= 0.046



Better: 0.5*0.3*0.6*0.6= 0.054



Search Strategies: Exhaustive Search

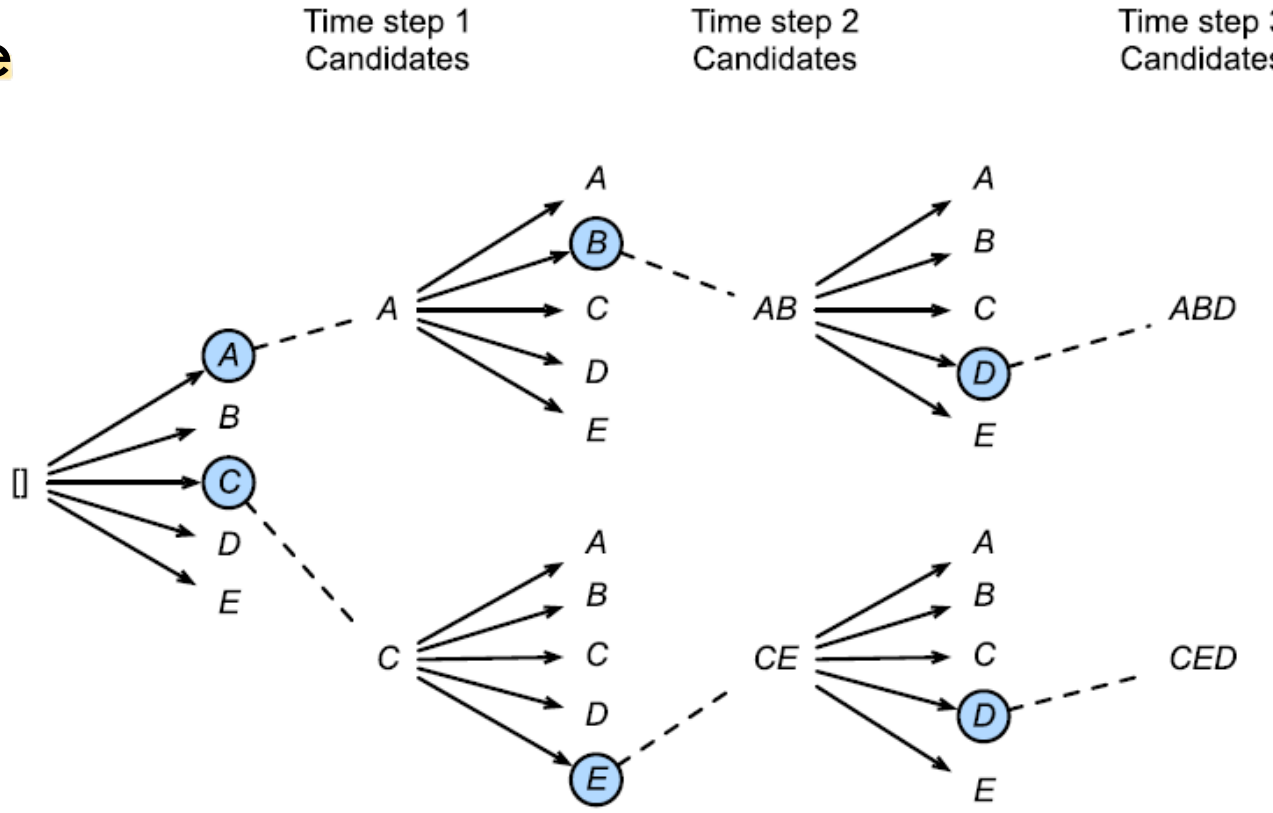
- Instead of finding just one sequence, enumerate all possible sequences and pick the best one
- This is not computational feasible $\mathcal{O}\left(|\mathcal{Y}|^{T'}\right)$
- For vacab size of 10,000 and sentence of length 10 requires

$$10000^{10} = 10^{40}$$

- Compared with greedy search $10000 \times 10 = 10^5$

Search Strategies: Beam Search

- Hyperparameter beam size k
- At each time step keep only the top k candidates
- For $k = 2$ candidates at each time step
  1. A, C
  2. AB, CE
  3. ABD, CED
  4. CEDA, CEDC
- Complexity: $\mathcal{O}(k|\mathcal{Y}|T')$



11 Attention Mechanisms and Transformers

Problem with Encoder-Decoder Architecture

- Encoding the meaning of the entire sentence using a single vector is difficult

- Is there a way to provide more information to the decoder?
- Cannot send a variable length sentence

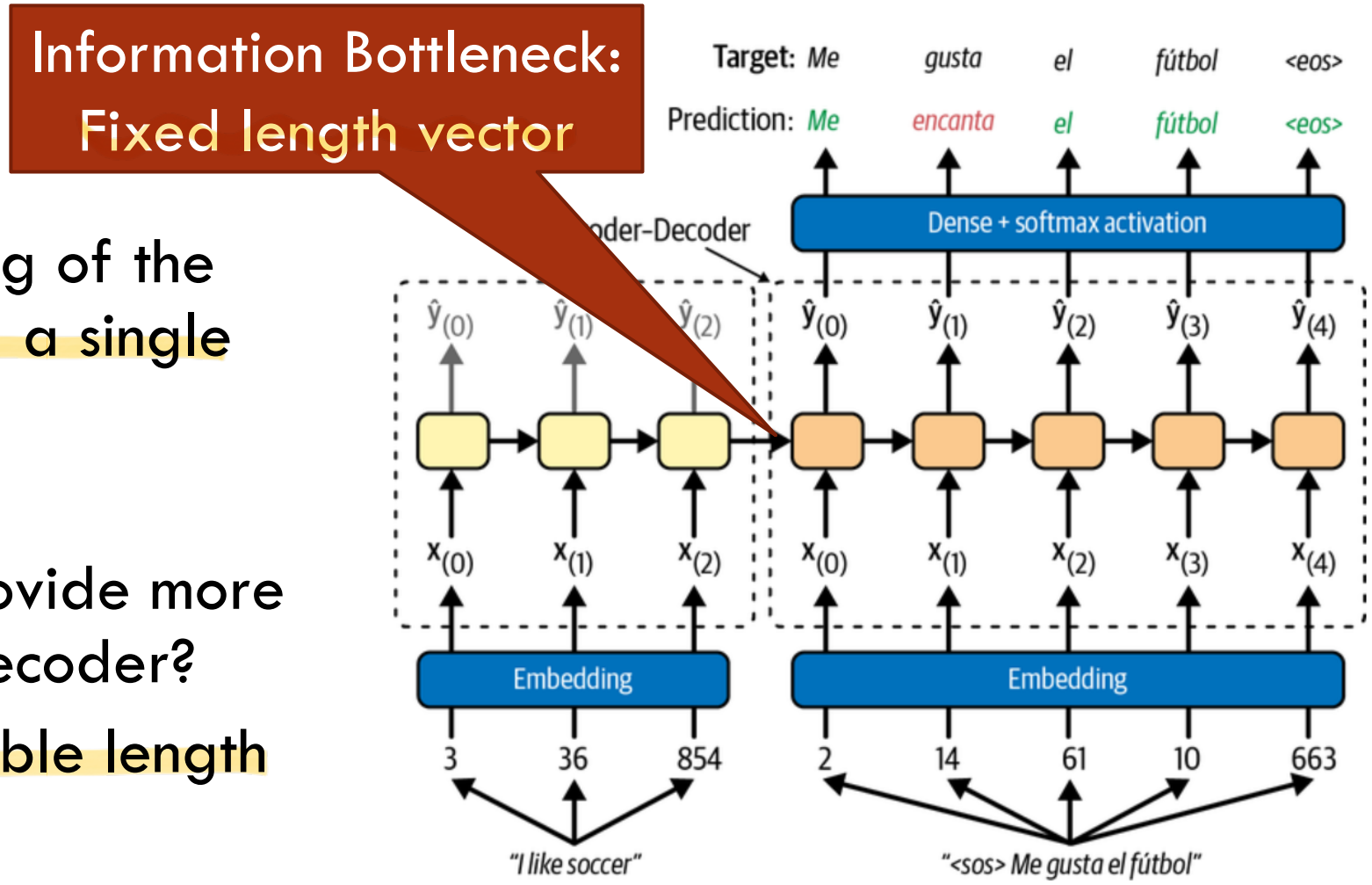


Figure 16-3. A simple machine translation model

Attention Mechanism Motivation

- Send another fixed length vector but in context
- Suppose decoder is in timestep 2
  - It has already generated "me gusta el"
  - Its internal state is h_2
  - Next it should generate "futbol"
- Send "soccer" to the decoder, i.e. send $\hat{y}_{(2)}$

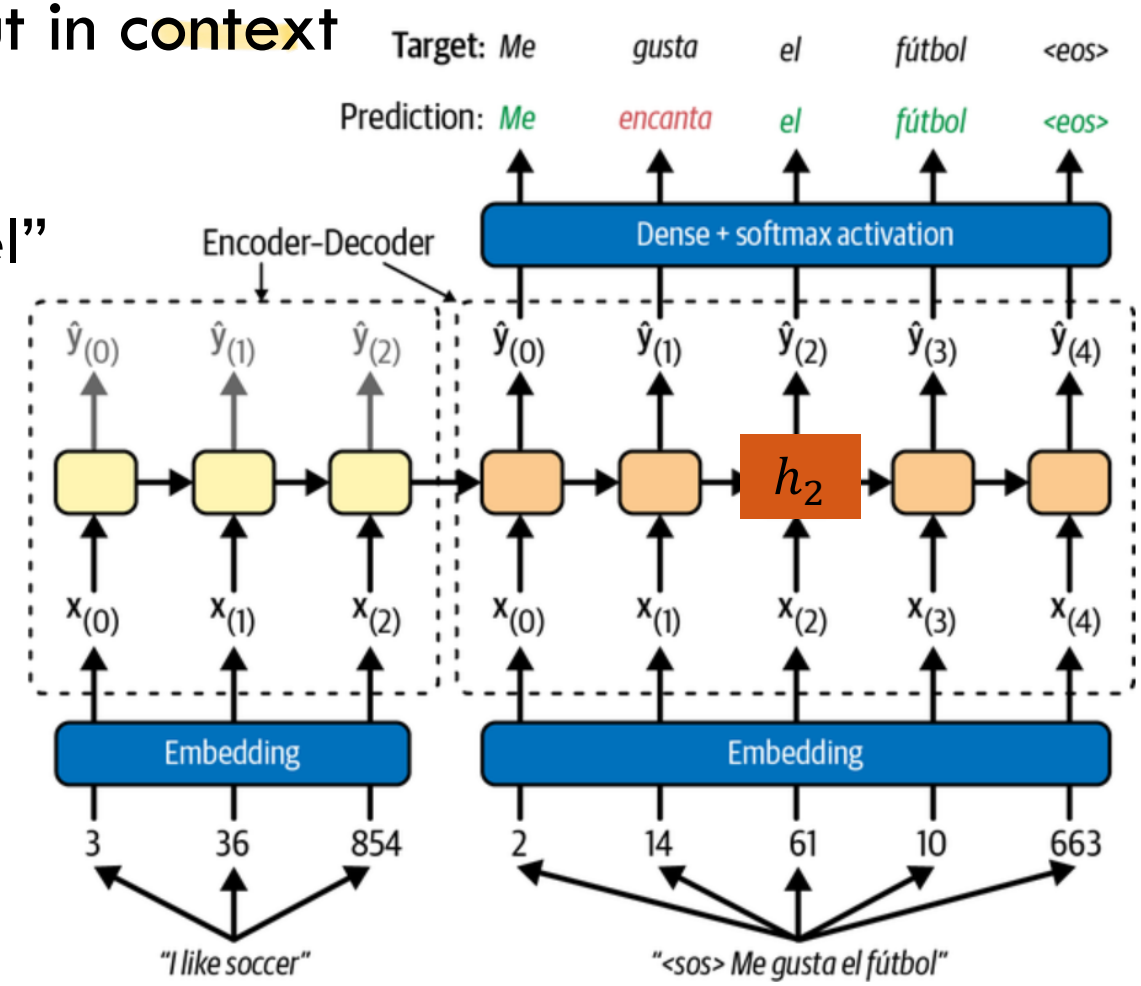


Figure 16-3. A simple machine translation model

Queries, Keys and Values

- How do we find the information to send to the decoder?
- Inspiration from databases, e.g., let the decoder query the encoder
- Suppose we have a database $\mathcal{D}$ of key value (k, v) pairs and query q

Key	Value
Zhang	Aston
Lipton	Zachary
Li	Mu
Smola	Alex
Hu	Rachael
Werness	Brent

Query: $q = "Li"$

Mu

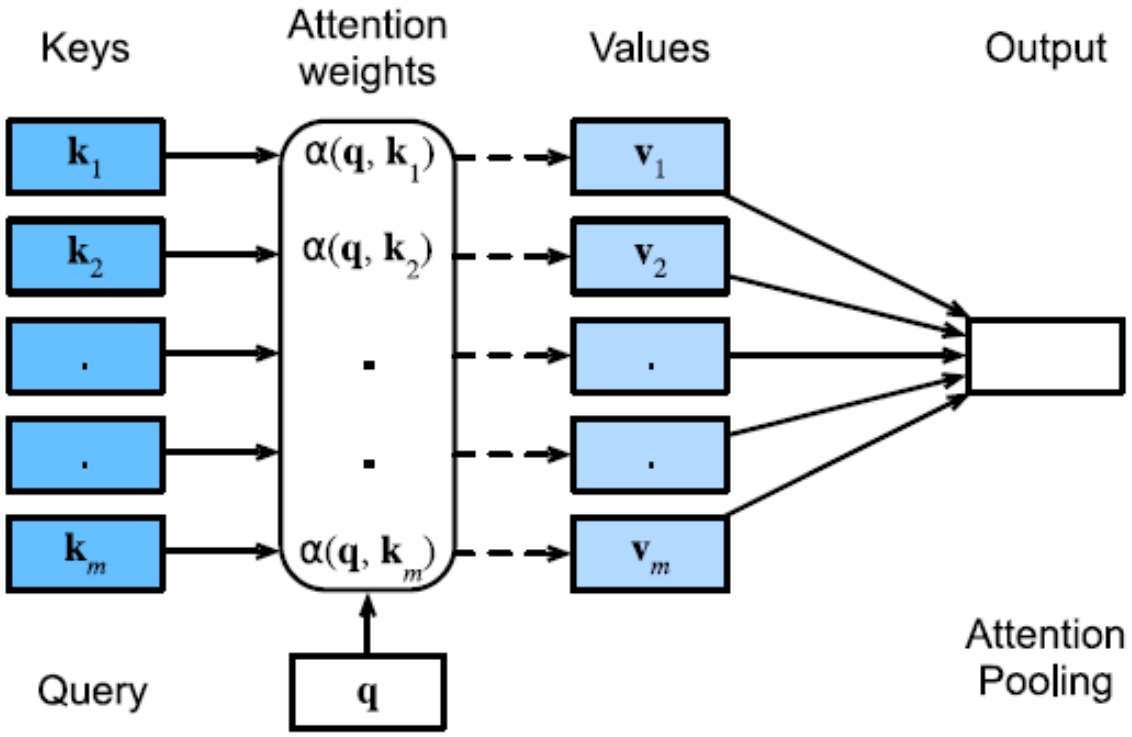
Zachary

Attention Mechanism (Bahdanau et al., 2014)

- Let $\mathcal{D} \equiv \{(\mathbf{k}_1, \mathbf{v}_1), \ldots, (\mathbf{k}_m, \mathbf{v}_m)\}$ be a database of m tuples of *keys* and *values*
- Let $\mathbf{q}$ be a *query*
- *Attention* over $\mathcal{D}$ is

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \equiv \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i$$

- Where $\alpha(\mathbf{q}, \mathbf{k}_i) \in \mathbb{R}$
- Also called *attention pooling*



Special Cases of Attention

- Attention is a linear combination of the values in the database
- Special cases by restricting the weights α
  - The weights are nonnegative.
  - The weights form a *convex combination*, i.e., weights are nonnegative and sum to 1.
  - Exactly one of the weights is one, and rest are 0.
  - All weights are equal $\left(\alpha = \frac{1}{m}\right)$, i.e., average pooling
- Convex combination is the most popular in deep learning,
  - E.g., apply softmax to the weights to make them nonnegative and sum to 1

Kernel Regression

- Kernel regression can be viewed through the lens of attention pooling by similarity
- In regression given a query $\mathbf{x}$, we want to compute the corresponding value $\mathbf{y}$
- In kernel regression, we only want to use training examples from a small neighborhood that is similar to $\mathbf{x}$,
- I.e., training examples with keys similar to the query

Similarity Kernels

- Common similarity kernels $\alpha(\mathbf{q}, \mathbf{k})$

$$\alpha(\mathbf{q}, \mathbf{k}) = \exp\left(-\frac{1}{2}\|\mathbf{q} - \mathbf{k}\|^2\right) \quad \text{Gaussian;}$$

$$\alpha(\mathbf{q}, \mathbf{k}) = 1 \text{ if } \|\mathbf{q} - \mathbf{k}\| \leq 1 \quad \text{Boxcar;}$$

$$\alpha(\mathbf{q}, \mathbf{k}) = \max(0, 1 - \|\mathbf{q} - \mathbf{k}\|) \quad \text{Epanechikov.}$$

- AKA *Parzen windows*

Kernel Regression

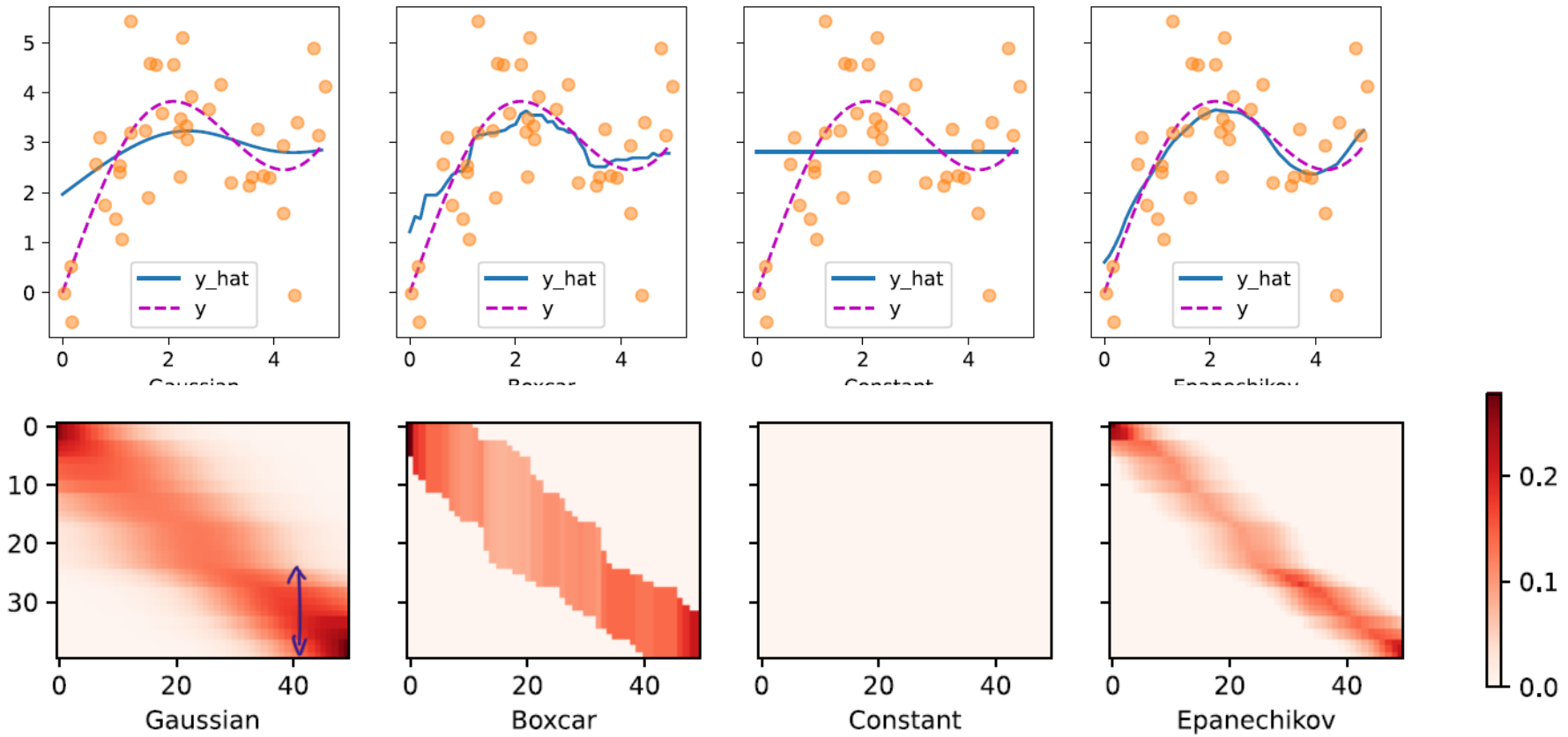
- To compute the regression value, we take weighted average:

$$f(\mathbf{q}) = \sum_i \mathbf{v}_i \frac{\alpha(\mathbf{q}, \mathbf{k}_i)}{\sum_j \alpha(\mathbf{q}, \mathbf{k}_j)}.$$

- AKA Nadaraya-Watson regression
- For classification the weights can be viewed as weighted votes

Notebook

- chapter_attention-mechanisms-and-transformers/attention-pooling.ipynb



Dot Product Attention

- Attention function based on distance squared

$$a(\mathbf{q}, \mathbf{k}_i) = -\frac{1}{2}\|\mathbf{q} - \mathbf{k}_i\|^2 = \mathbf{q}^\top \mathbf{k}_i - \frac{1}{2}\|\mathbf{k}_i\|^2 - \frac{1}{2}\|\mathbf{q}\|^2.$$

- Notice
  - The last term only depends on the query. It can be dropped.
  - If the keys are normalized, then it can be dropped as well
- Then only the dot product is left
- Luong *et al.*, 2015

Scaled Dot Product Attention

- If the query $q \in \mathbb{R}^d$ and $k_i \in \mathbb{R}^d$ are iid drawn with zero mean and unit variance, then the result dot product have variance d
- To keep the dot product unit variance, we multiple by $1/\sqrt{d}$

$$a(\mathbf{q}, \mathbf{k}_i) = \mathbf{q}^\top \mathbf{k}_i / \sqrt{d}.$$

- Then, we normalize as well

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \text{softmax}(a(\mathbf{q}, \mathbf{k}_i)) = \frac{\exp(\mathbf{q}^\top \mathbf{k}_i / \sqrt{d})}{\sum_{j=1} \exp(\mathbf{q}^\top \mathbf{k}_j / \sqrt{d})}$$

- Vaswani, *et al.*, 2017

Scaled Dot Product Attention

- For querying with minibatches of n queries against m key-value pairs, where the queries and keys have length d, and values have length v
- Let queries $\mathbf{Q} \in \mathbb{R}^{n \times d}$, keys $\mathbf{K} \in \mathbb{R}^{m \times d}$, and values $\mathbf{V} \in \mathbb{R}^{m \times v}$

$$\text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \in \mathbb{R}^{n \times v}$$

n x m

m x v

n x v

Additive Attention

- Given a query $\mathbf{q} \in \mathbb{R}^q$ and a specific $\mathbf{k} \in \mathbb{R}^k$ ($\mathbf{q}, \mathbf{k}$ different sizes)

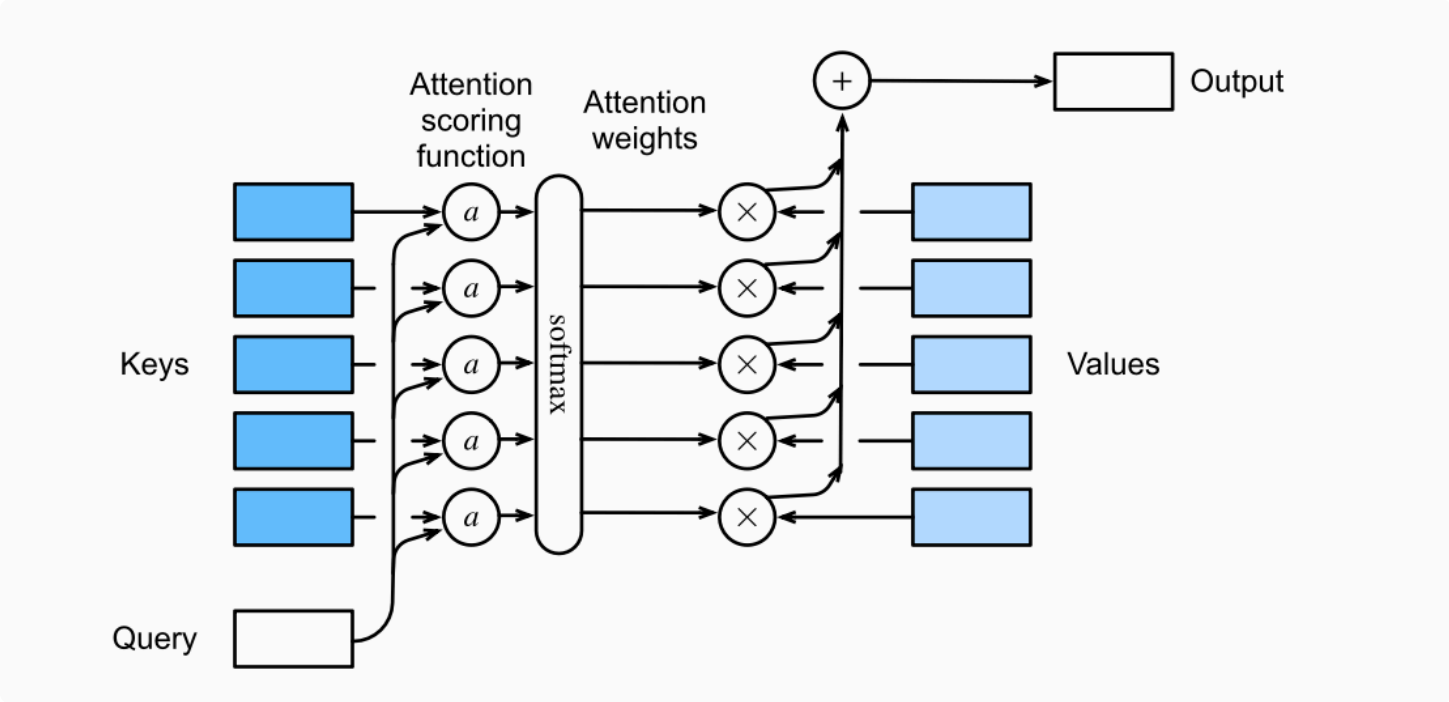
$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^\top \tanh(\mathbf{W}_q \mathbf{q} + \mathbf{W}_k \mathbf{k}) \in \mathbb{R}$$

  With learnable parameters $\mathbf{W}_q \in \mathbb{R}^{h \times q}, \mathbf{W}_k \in \mathbb{R}^{h \times k}$ and $\mathbf{w}_v \in \mathbb{R}^h$

- Attention function originally proposed by Bahdanau et al., 2014
- Aka Bahdanau attention

Notebook

- chapter_attention-mechanisms-and-transformers/attention-scoring-functions.ipynb



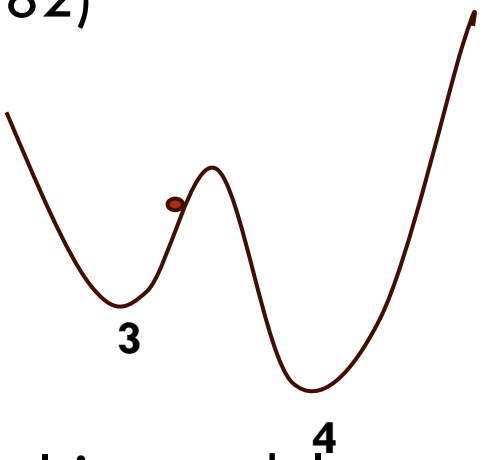
Hopfield Network

- Proposed by John Hopfield, 2024 Nobel Prize for Physics
- A type of energy-based recurrent neural network (proposed in 1982)

$$E = -\frac{1}{2}\sum_{i,j} w_{ij}x_i x_j - \sum_i b_i x_i = -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$$

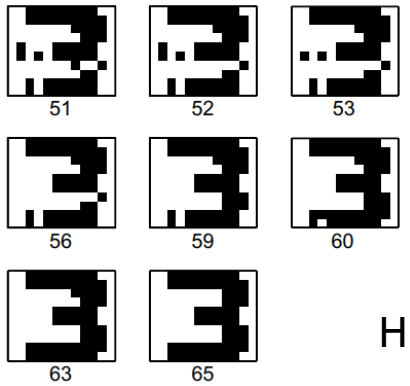
  - Find optimum state by gradient descent, $x_k = 0 \rightarrow x_k = 1, x_k \in \{0,1\}$

$$\Delta E_k = -\sum_i w_{ki}x_i - b_k$$



- Also known associative memory, or content addressable memory, or Ising model

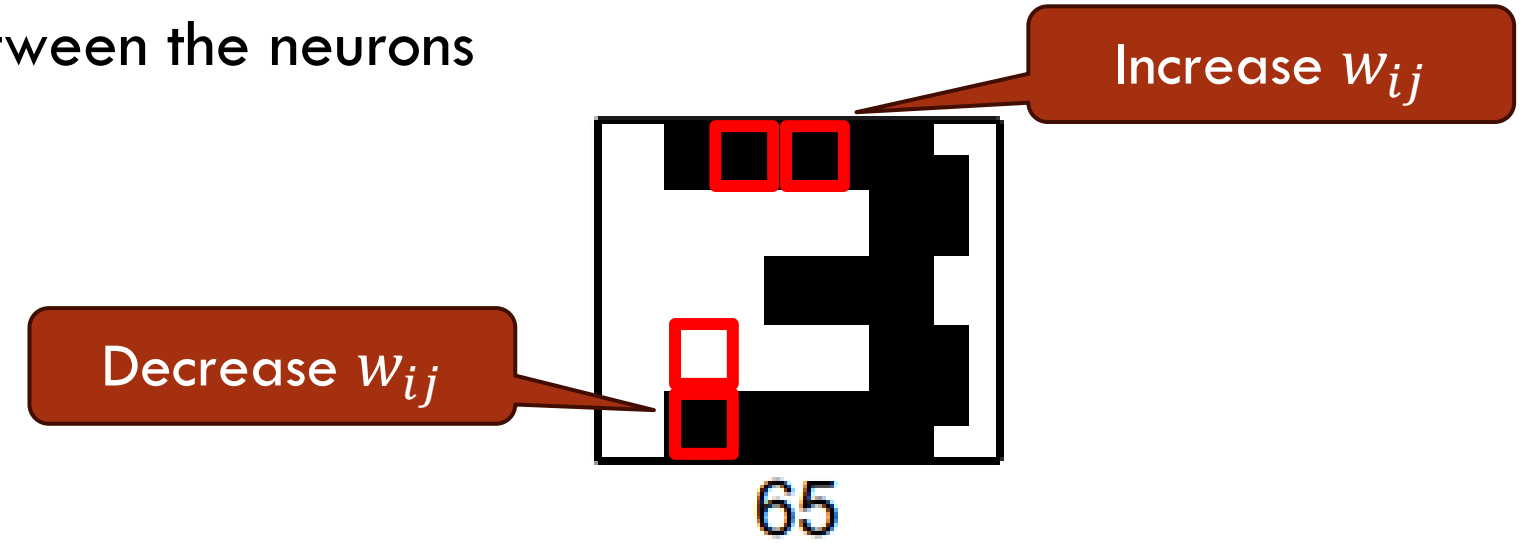
Use fuzzy image to search:



H Ma et al., 2014

Training Hopfield Networks

- Hebbian learning:
  - If two neurons tend fire together, increase weights between the weight between the neurons



Nobel Prize Press Release

- *They trained artificial neural networks using physics.*
- *John Hopfield created an associative memory that can store and reconstruct images and other types of patterns in data.*
- *John Hopfield invented a network that uses a method for saving and recreating patterns. We can imagine the nodes as pixels. The **Hopfield network** utilises physics that describes a material's characteristics due to its atomic spin – a property that makes each atom a tiny magnet. The network as a whole is described in a manner equivalent to the energy in the spin system found in physics, and is trained by finding values for the connections between the nodes so that the saved images have low energy. When the Hopfield network is fed a distorted or incomplete image, it methodically works through the nodes and updates their values so the network's energy falls. The network thus works stepwise to find the saved image that is most like the imperfect one it was fed with.*

Problems with Original Hopfield Network

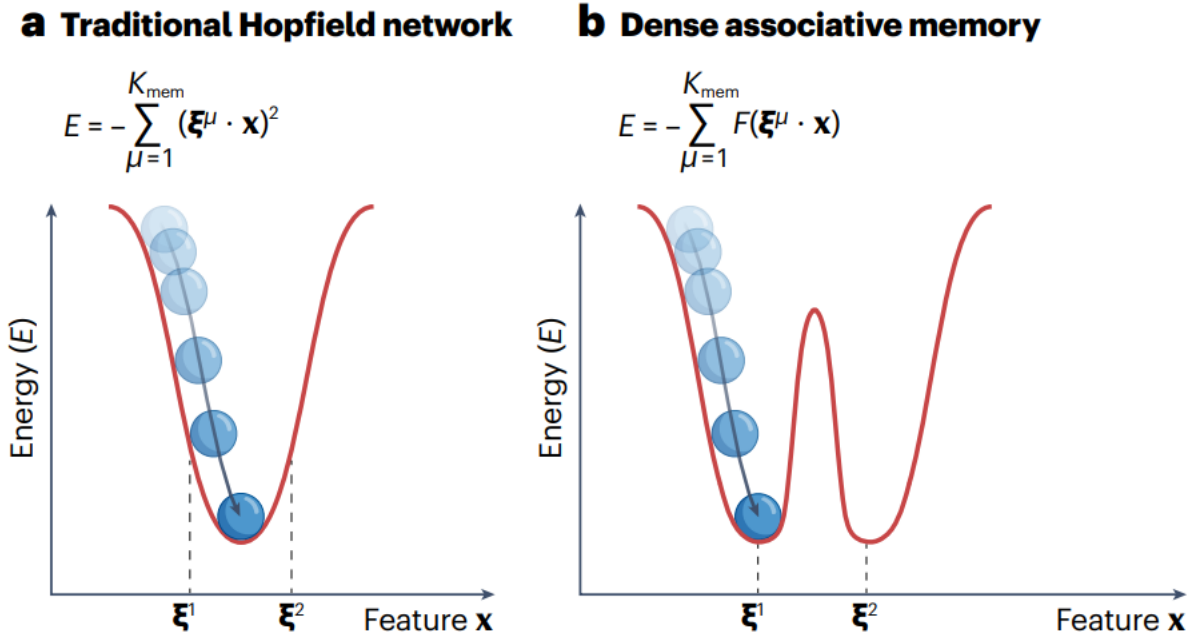
- Neurons x_i are discrete, either {0, 1} or {-1, 1}
- Can only stored limited number of patterns in the network
  - If try to store too many patterns, then network may return result that is a mixture of multiple patterns
  - Can only store linear number of patterns
  - For $\boldsymbol{x} \in \mathcal{R}^N$, storage capacity is $\sim 0.138N$ approximate patterns

Developments in Hopfield Network

- Extend neurons to continuous variables
- Dense associative memory (Krotov & Hopfield, 2016)
  - Store exponential number of patterns $\sim c^{\frac{N-1}{4}}$, for some constant c
  - Large basins of attraction for patterns
  - Use new energy equation with rapidly growing activation function $F()$:

$$E = -\sum_{\mu=1}^{K_{mem}} F(\xi^{\mu} \cdot x)$$

    - ξ^{μ} is the pattern number μ
    - K_{mem} is the number of patterns



Krotov (2023), A new frontier for Hopfield networks

Developments in Hopfield Network

- "*Hopfield networks is all you need*," Ramsauer et al, 2021
- Attention is like associative memory
- Let the associative memory patterns be the attention keys
- If derivative of activation function $F()$ is the sigmoid, Hopfield network acts like the attention mechanism
- "*Energy Transformers*," Hoover et al, NeurIPS 2023