6.7900 Machine Learning (Fall 2024)

Lecture 13: transformers (cont'd), generalization

(supporting slides)

Outline

- Modeling sequences with state space models: Recurrent Neural Networks (**RNNs**)
- Modeling images (and related data): Convolutional Neural Networks (**CNNs**)
- Modeling graphs, data on graphs: Graph Neural Networks (**GNNs**)
- Modeling sequences, images, etc with **Transformers**
- Small steps towards generalization, uniform convergence, (algorithmic stability)

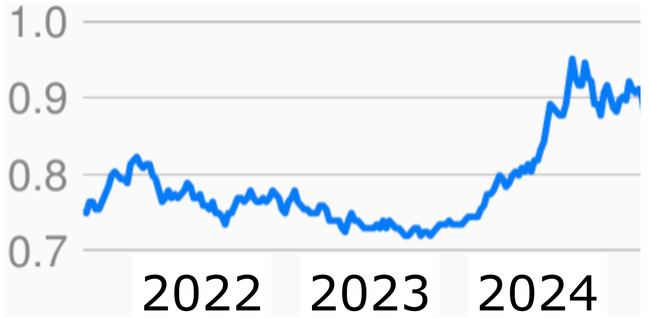
Transformers: broadly applicable architectures

Language modeling, translation, Q/A

The next revolutionary AI startup is […]

Time sequence prediction (events, health trajectories, etc)

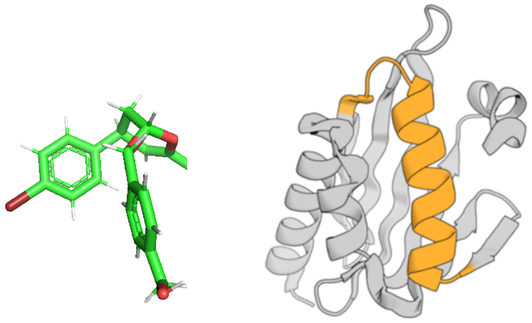
Time series prediction



?

Image representation, prediction, generation

Drug design, molecule modeling



Learning to control

Why do we care about transformers?

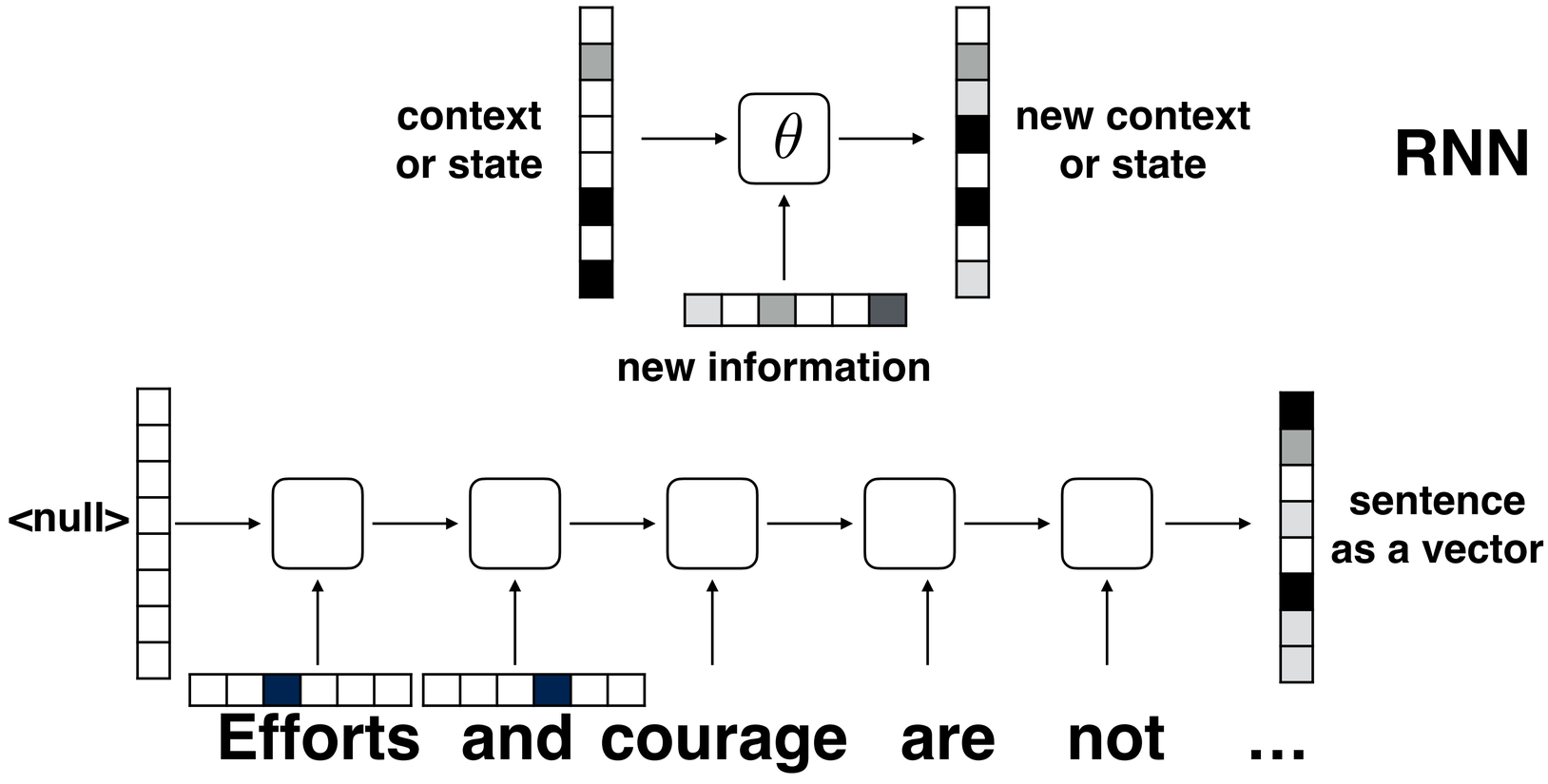
- It has become the dominant neural architecture
  - natural language text (Bert, T5, GPT, etc)
  - biosequences, molecules (e.g., protein models)
  - images (e.g., vision transformer)
  - graphs, etc.

- What’s so good about transformers?
  - easier access to distal dependencies
  - longer input sequences do not need more parameters (repeated architecture)
  - more parallelizable than recurrent neural models (LSTMs)
  - fewer iterative steps (easier to learn)
  - lots of easy pre-training tasks for large models

- But make sure you have some compute power …

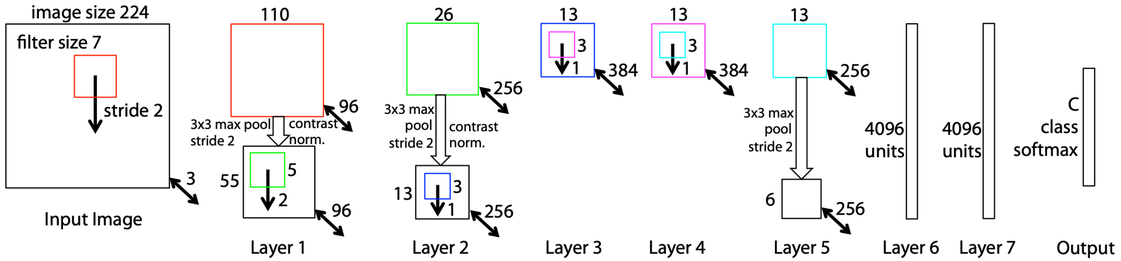
Why not RNNs?

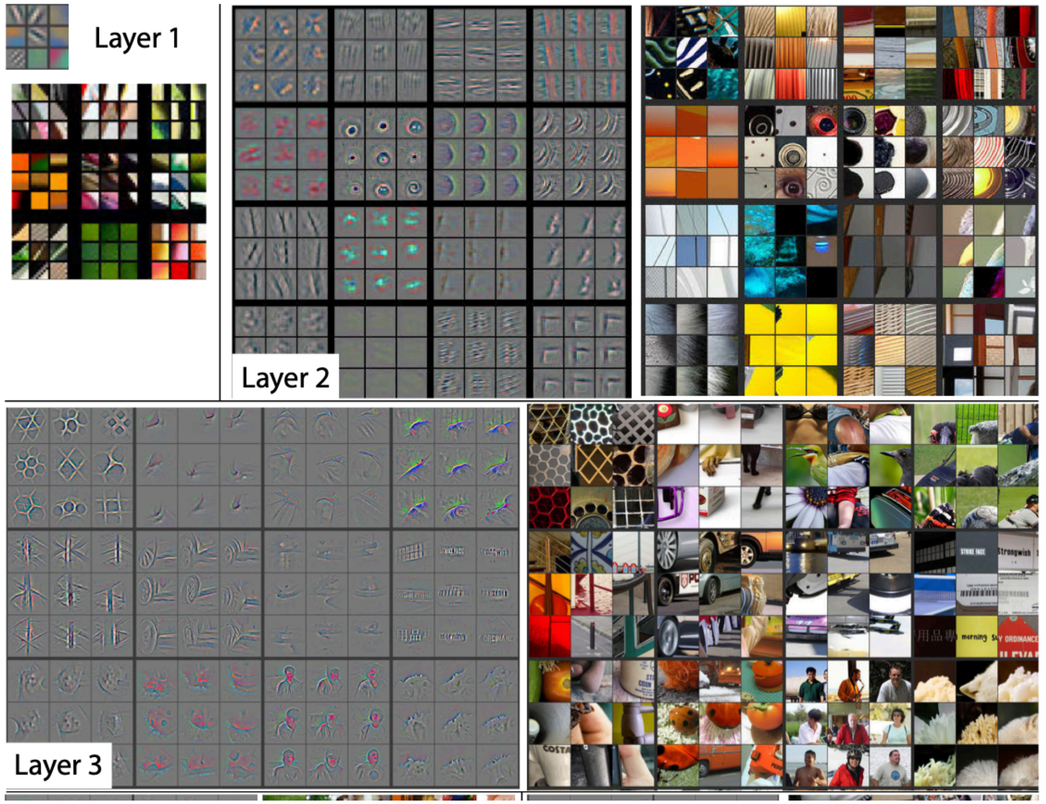
- Recurrent neural networks (RNNs) process sentences iteratively, sequentially — we cannot easily parallelize this type of computation
- RNNs have to compress all the context (past information) into a (single) vector
- E.g., RNN encoder of a sentence



CNNs and locality

- CNN is built from local “kernels” that look for (local) features, then combine these at later layers
- The kernels are inherently local, would require lots of parameters in order for us to extend these features to over a larger part of the image/sequence





[Zeiler et al. 2013]

“Attention is all you need”…

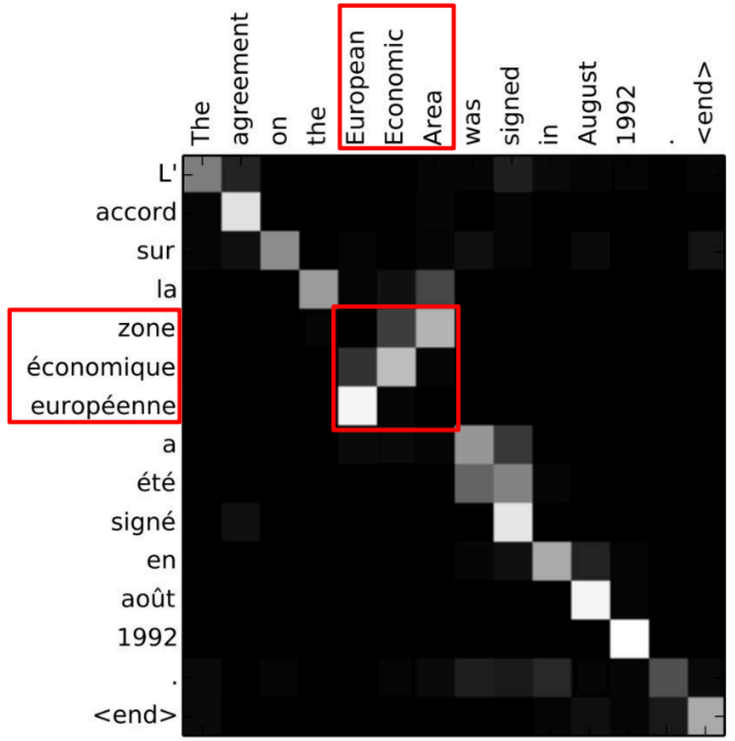
- One of the key ideas in transformers is the “attention mechanism” that lets the model focus on the relevant parts of the input (and intermediate) signals.

Preface: attention in machine translation

- Attention was introduced in (early) neural machine translation architectures
- E.g., English to French translation

- **Input sentence:** *"The agreement on the European Economic Area was signed in August 1992"*

- **Output sentence:** *"L'accord sur la zone économique européenne a été signé en août 1992"*



attention matrix during translation

[Bahdanau et al. 2015]

Preface: attention mechanism

- Older architectures for captioning mapped an image to a (fixed length) vector and then this vector to the corresponding caption words
- We can instead require the model to "attend to" relevant parts of the image in the process of creating each caption word

A woman is throwing a frisbee in a park.

A group of people sitting on a boat in the water.

- Not always perfect…

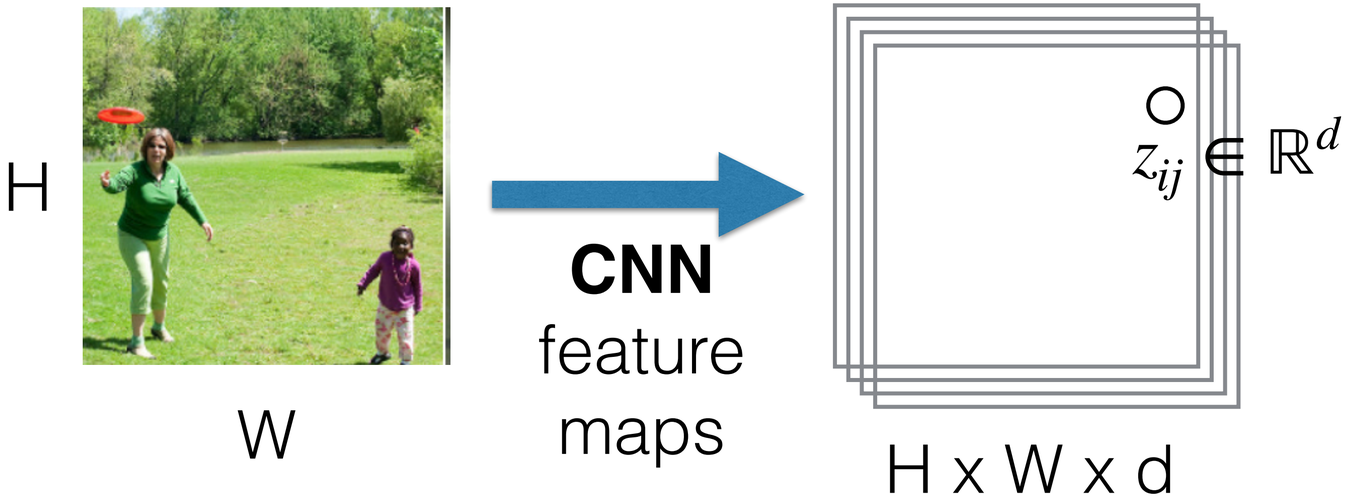
A large white bird standing in a forest.

[Xu et al. 2015]

Preface: attention mechanism

- Consider the step before we select the word “frisbee”
- The earlier part of the sentence (“A woman is throwing a”) summarized as $h \in \mathbb{R}^d$

A woman is throwing a frisbee in a park.



Preface: attention mechanism

- Consider the step before we select the word “frisbee”
- The earlier part of the sentence (“A woman is throwing a”) summarized as $h \in \mathbb{R}^d$

A woman is throwing a frisbee in a park.

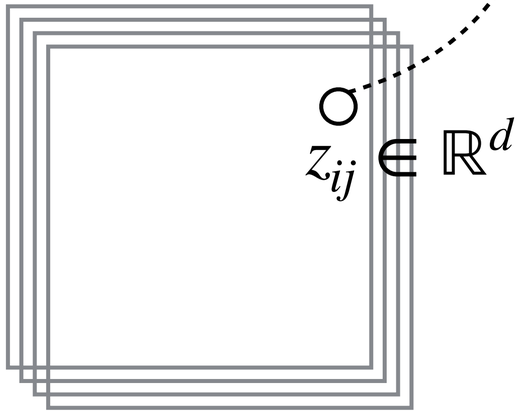
score each position relative to h

$$s_{ij} = \text{score}(h, z_{ij})$$

H

W

CNN feature maps



$z_{ij} \in \mathbb{R}^d$

H x W x d

Preface: attention mechanism

- Consider the step before we select the word “frisbee”
- The earlier part of the sentence (“A woman is throwing a”) summarized as $h \in \mathbb{R}^d$

A woman is throwing a frisbee in a park.

score each position relative to h

$$s_{ij} = \text{score}(h, z_{ij})$$

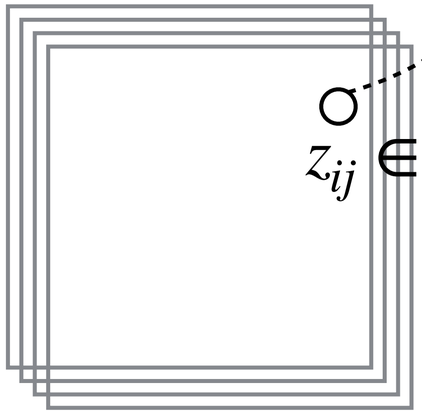
attention weights (normalized)

$$a_{ij} = \frac{e^{s_{ij}}}{\sum_{kl} e^{s_{kl}}}$$

H

W

CNN feature maps



$z_{ij} \in \mathbb{R}^d$

H x W x d

Preface: attention mechanism

- Consider the step before we select the word “frisbee”
- The earlier part of the sentence (“A woman is throwing a”) summarized as $h \in \mathbb{R}^d$

A woman is throwing a frisbee in a park.

score each position relative to h

$$s_{ij} = \text{score}(h, z_{ij})$$

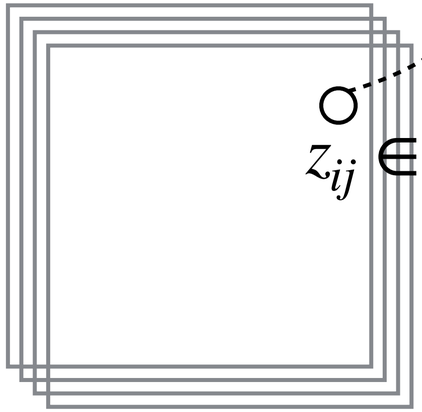
attention weights (normalized)

$$a_{ij} = \frac{e^{s_{ij}}}{\sum_{kl} e^{s_{kl}}}$$

H

W

CNN feature maps



$z_{ij} \in \mathbb{R}^d$

H x W x d

H x W

Preface: attention mechanism

- Consider the step before we select the word “frisbee”
- The earlier part of the sentence (“A woman is throwing a”) summarized as $h \in \mathbb{R}^d$

A woman is throwing a frisbee in a park.

score each position relative to h

$$s_{ij} = \text{score}(h, z_{ij})$$

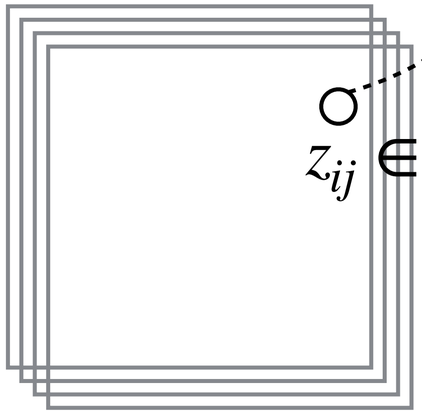
attention weights (normalized)

$$a_{ij} = \frac{e^{s_{ij}}}{\sum_{kl} e^{s_{kl}}}$$

H

W

CNN feature maps



$z_{ij} \in \mathbb{R}^d$

H x W x d

$\odot$

H x W

weighted aggregation

$$c = \sum_{ij} a_{ij} z_{ij}$$

d x 1

Preface: attention mechanism

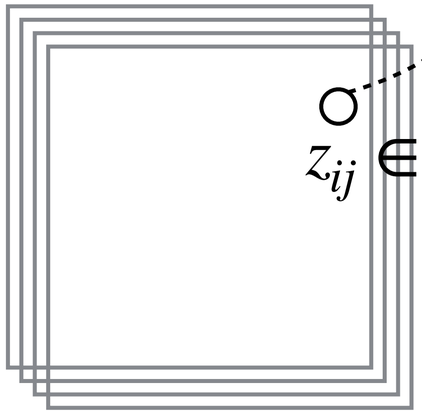
- Consider the step before we select the word “frisbee”
- The earlier part of the sentence (“A woman is throwing a”) summarized as $h \in \mathbb{R}^d$

A woman is throwing a frisbee in a park.

H

W

CNN feature maps



$z_{ij} \in \mathbb{R}^d$

H x W x d

score each position relative to h

$$s_{ij} = \text{score}(h, z_{ij})$$

attention weights (normalized)

$$a_{ij} = \frac{e^{s_{ij}}}{\sum_{kl} e^{s_{kl}}}$$

H x W

weighted aggregation

$$c = \sum_{ij} a_{ij} z_{ij}$$

“frisbee”

h

d x 1

Preface: sentiment with attention

- A review can be assessed in multiple ways (different aspects), such as **food**, price, service, ambience, etc.
- We would like the method to “attend to”, i.e., highlight different parts of the review when classifying aspects as positive or negative

Great food in a restaurant wthout too much music/noice

Not cheap, but the food is excellent (roast pork my favorite). Waiters are attentive, and after it turned out that the kitchen was sort of slow in cooking our dinner, the waiter provided a nice free dessert. We had not even complained.

Preface: a retrieval view of "hard" attention

- Consider a query q (a vector) against a database of keys and values
- Each database value v (a vector) is associated with a key k (also a vector) for retrieval purposes

E.g., $\begin{cases} k_i = x_i \\ v_i = x_i \end{cases}$

here keys and values are just the word vectors

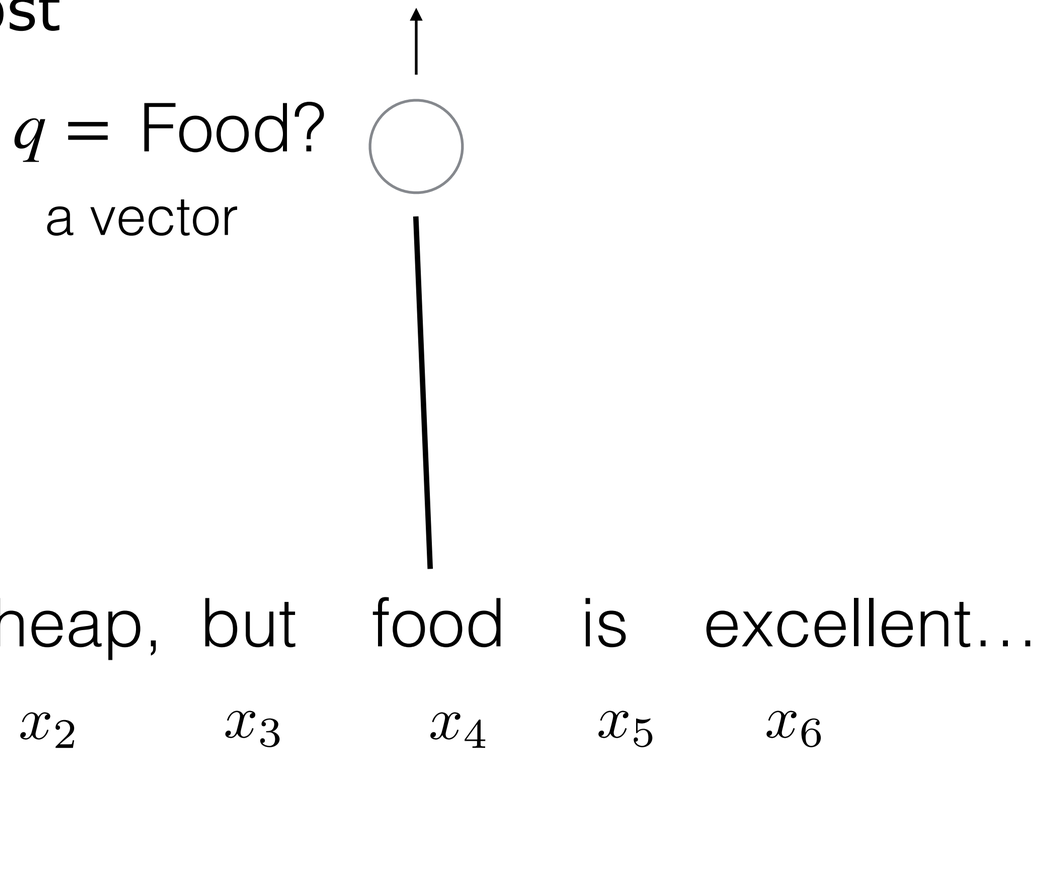
Not	cheap,	but	food	is	excellent…
x_1	x_2	x_3	x_4	x_5	x_6

word vector

Preface: a retrieval view of "hard" attention

- Consider a query q (a vector) against a database of keys and values
- Each database value v (a vector) is associated with a key k (also a vector) for retrieval purposes
- **Step 1:** find the key most similar to the query

$$\hat{i} = \text{argmax}_i \{ q \cdot k_i \}$$



E.g., $\begin{cases} k_i = x_i \\ v_i = x_i \end{cases}$

here keys and values are just the word vectors

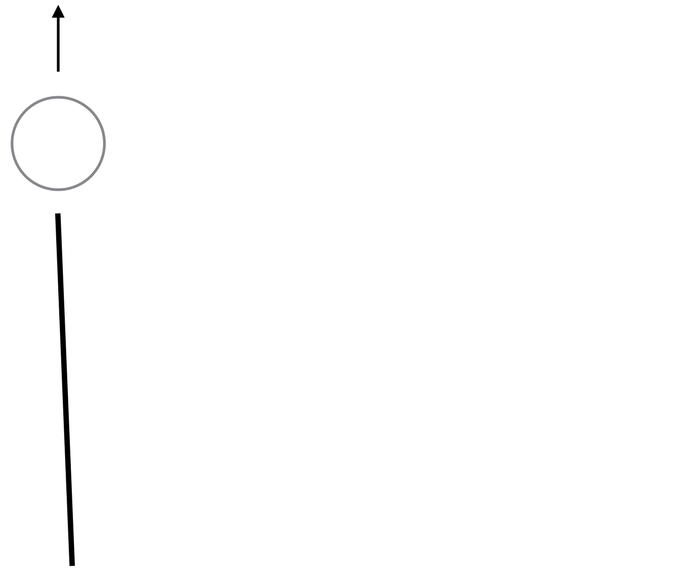
Preface: a retrieval view of "hard" attention

- Consider a query q (a vector) against a database of keys and values
- Each database value v (a vector) is associated with a key k (also a vector) for retrieval purposes
- **Step 1:** find the key most similar to the query

$$\hat{i} = \text{argmax}_i \{ q \cdot k_i \}$$

- **Step 2:** return the associated value

$$\text{output} = v_{\hat{i}}$$

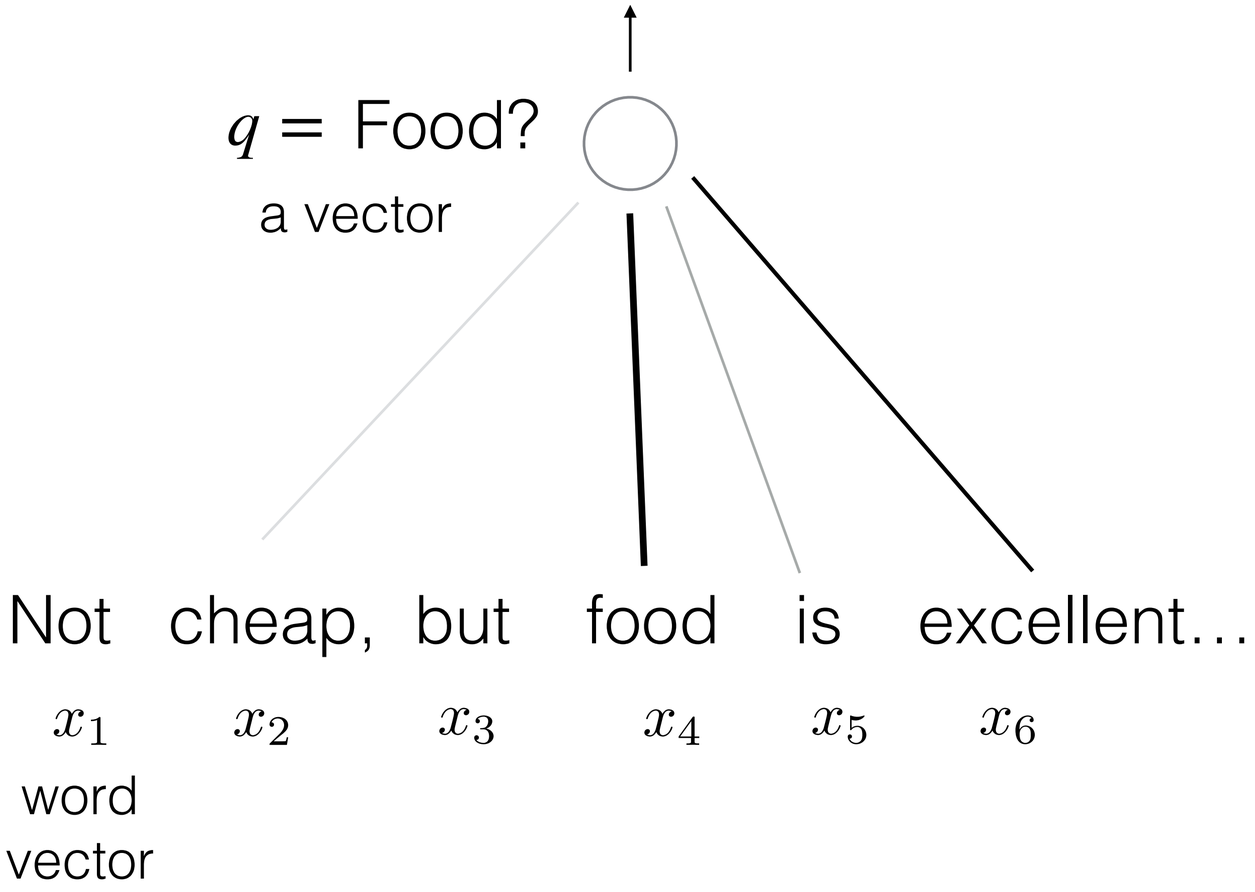


E.g., $\begin{cases} k_i = x_i \\ v_i = x_i \end{cases}$

here keys and values are just the word vectors

Preface: attention mechanism

- Focused sentiment prediction with simple attention



E.g., $\begin{cases} k_i = x_i \\ v_i = x_i \end{cases}$

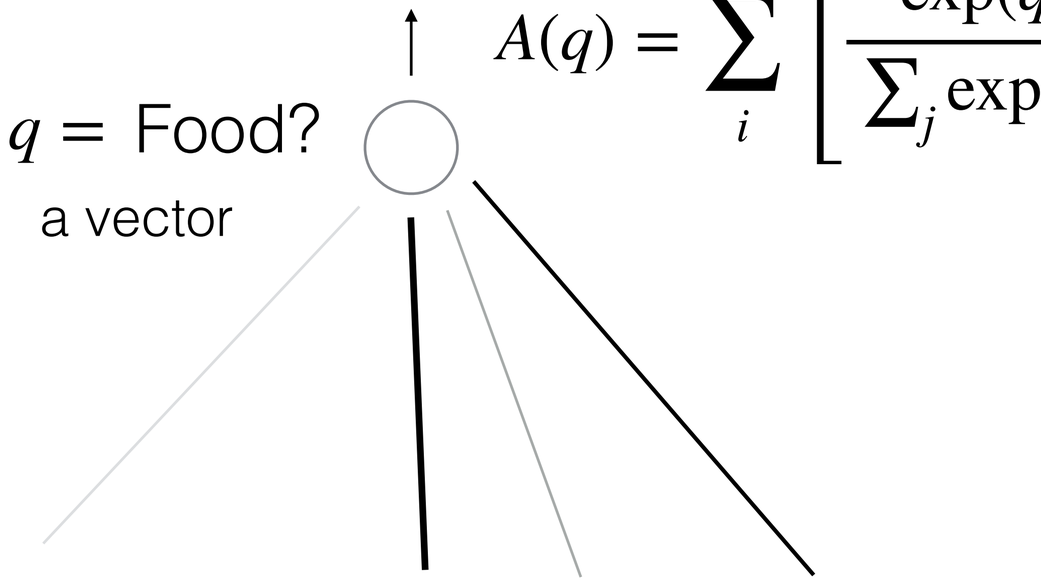
here keys and values are just the word vectors

Preface: attention mechanism

- Focused sentiment prediction with simple attention

weighted (focused) average of word vectors

$$A(q) = \sum_i \left[\frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_j)} \right] v_i$$

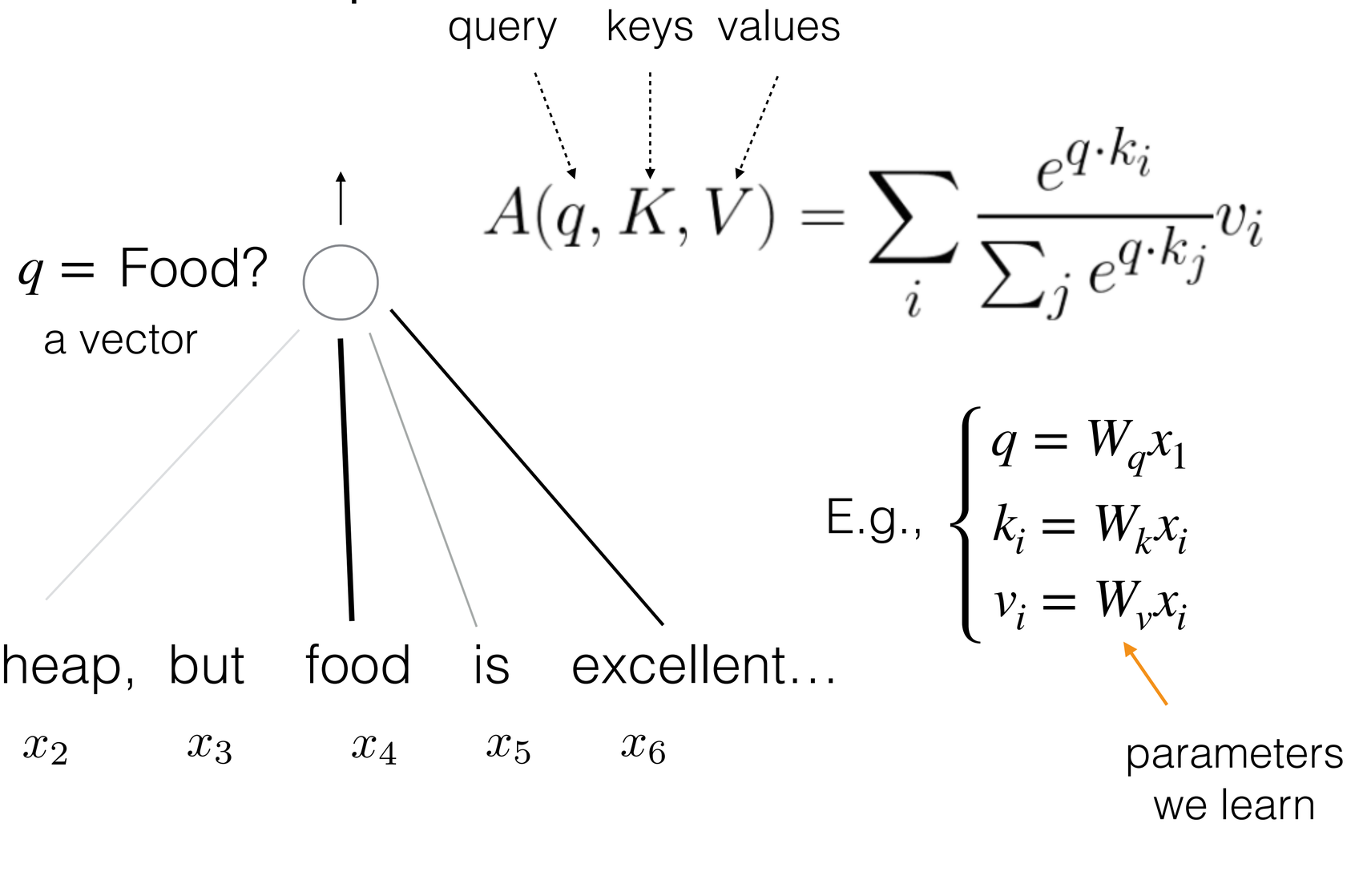


E.g., $\begin{cases} k_i = x_i \\ v_i = x_i \end{cases}$

here keys and values are just the word vectors

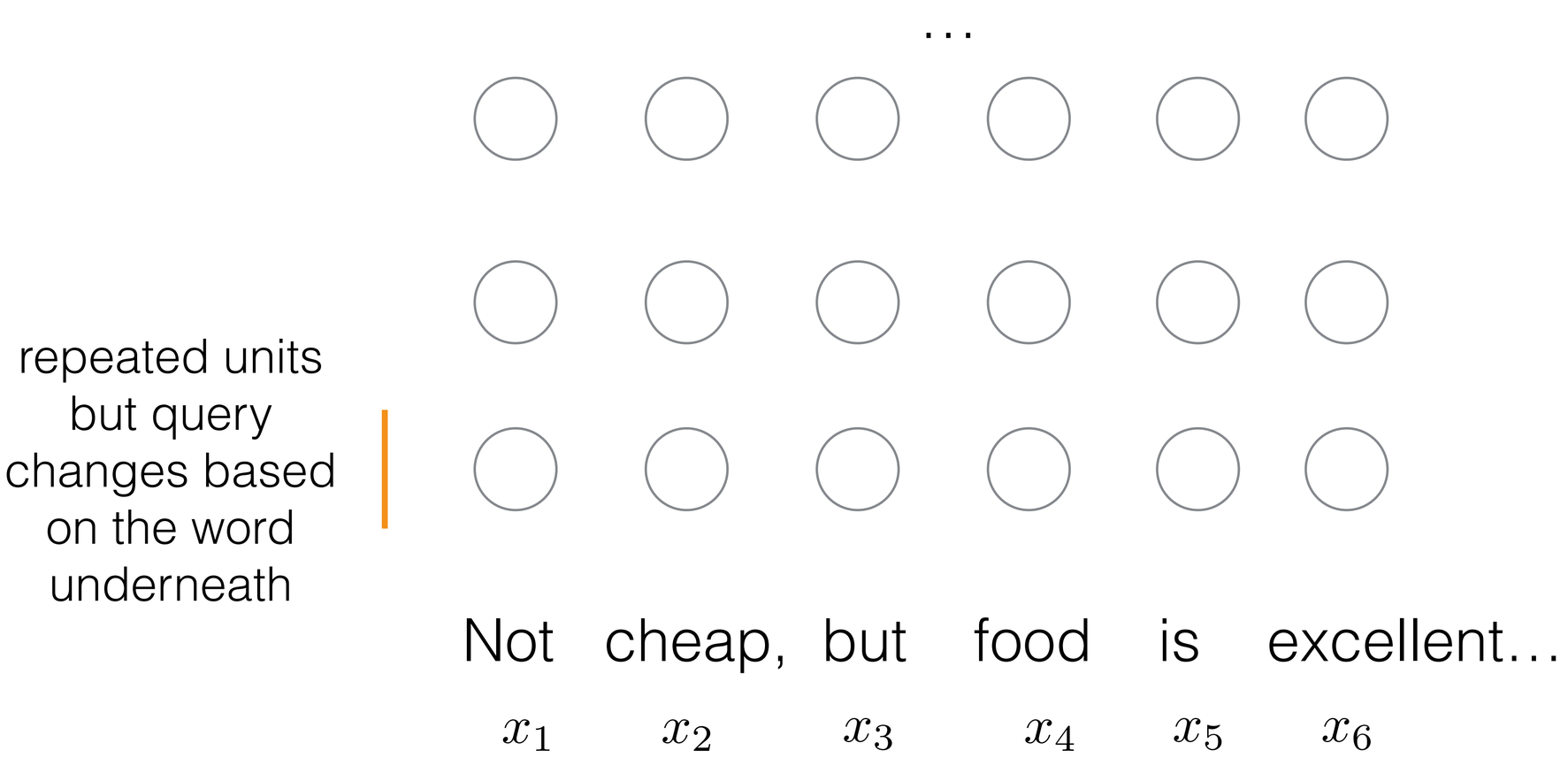
Preface: attention mechanism with parameters

- Focused sentiment prediction with simple attention



Transformers

- A transformer network involves multiple layers, multi-head (self-)attention



Transformer attention

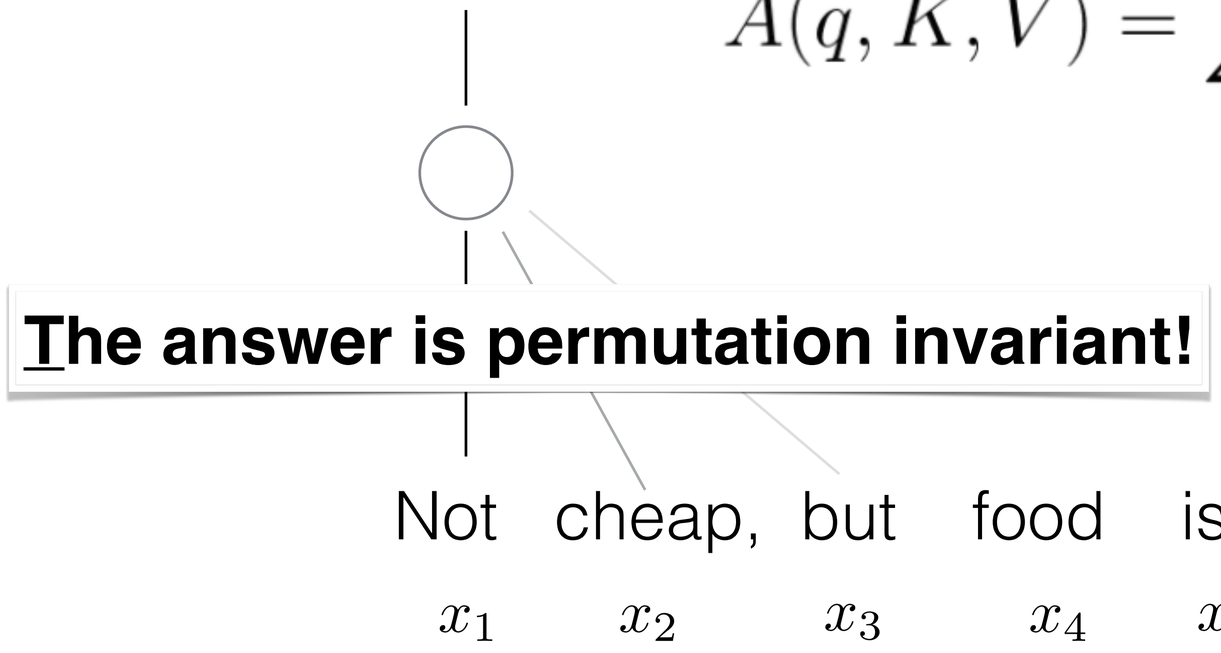
- A modified (self-)attention mechanism

query keys values

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

parameters we learn

E.g., $\begin{cases} q = W_q x_1 \\ k_i = W_k x_i \\ v_i = W_v x_i \end{cases}$



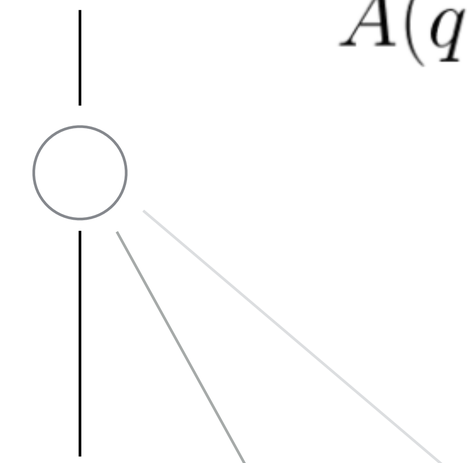
[Vaswani et al. 2017]

Transformer attention

- A modified (self-)attention mechanism with positional encoding

query keys values

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$



E.g., $\begin{cases} q = W_q[x_1 + pos_1] \\ k_i = W_k[x_i + pos_i] \\ v_i = W_v[x_i + pos_i] \end{cases}$

Not cheap, but food is excellent…

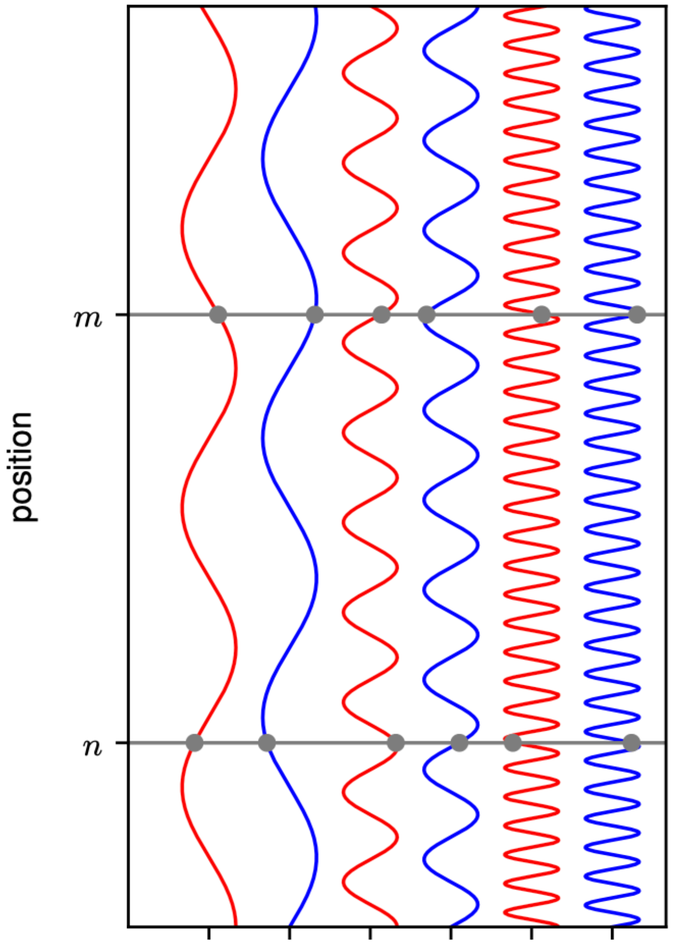
x_1 x_2 x_3 x_4 x_5 x_6

pos_1 pos_2 pos_3 pos_4 pos_5 pos_6

[Vaswani et al. 2017]

Positional encoding

- We can use values of trigonometric functions (sin's and cos's) with different wavelengths to translate sequence positions (j=0,1,2,…) into vectors



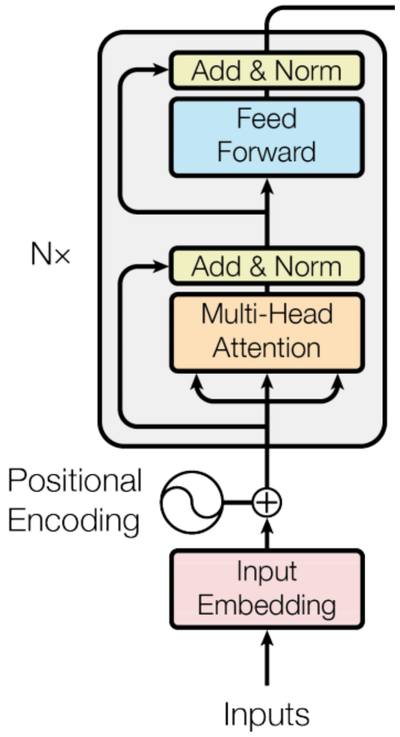
$$pos_n(i) = \begin{cases} \sin\left(\frac{n}{L^{i/D}}\right), & \text{if } i \text{ is even,} \\ \cos\left(\frac{n}{L^{(i-1)/D}}\right), & \text{if } i \text{ is odd.} \end{cases}$$

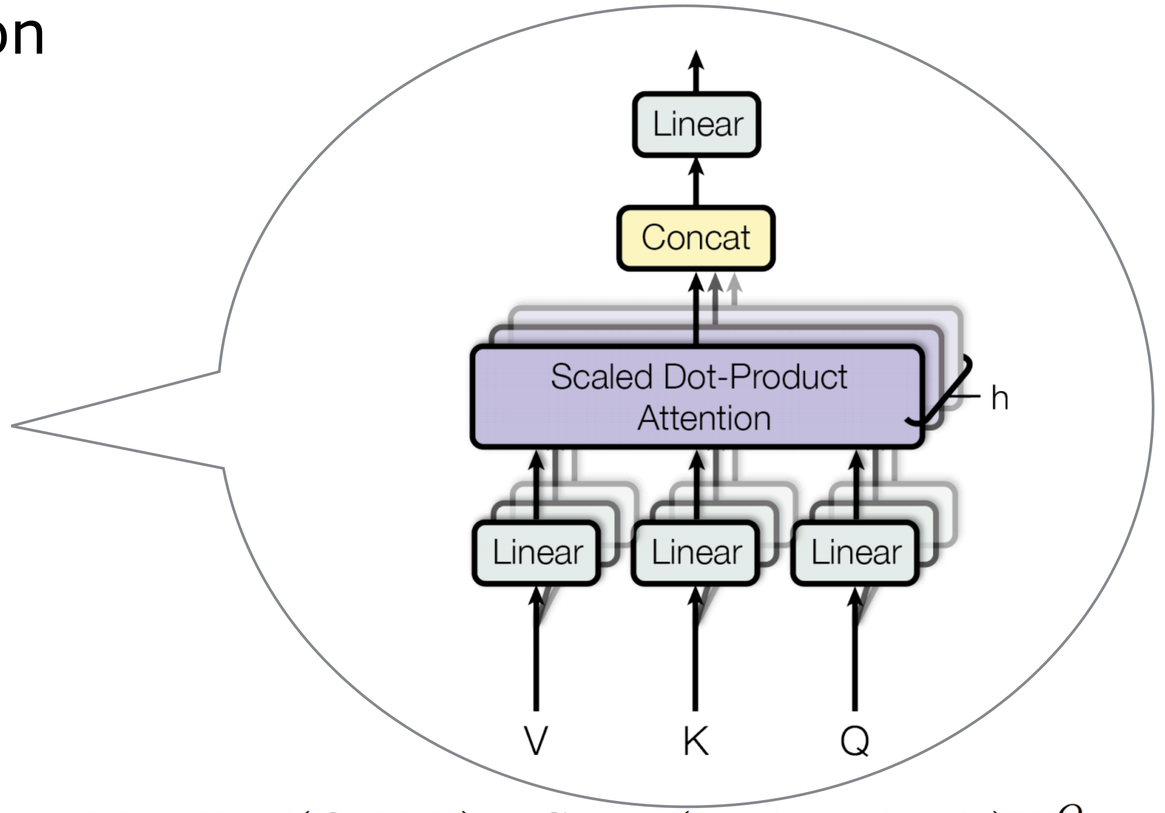
pos_n a positional vector for the nth position in the sequence

[figure adapted from Bishop et al.]

A transformer unit

- Multiple layers, multi-head attention





$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

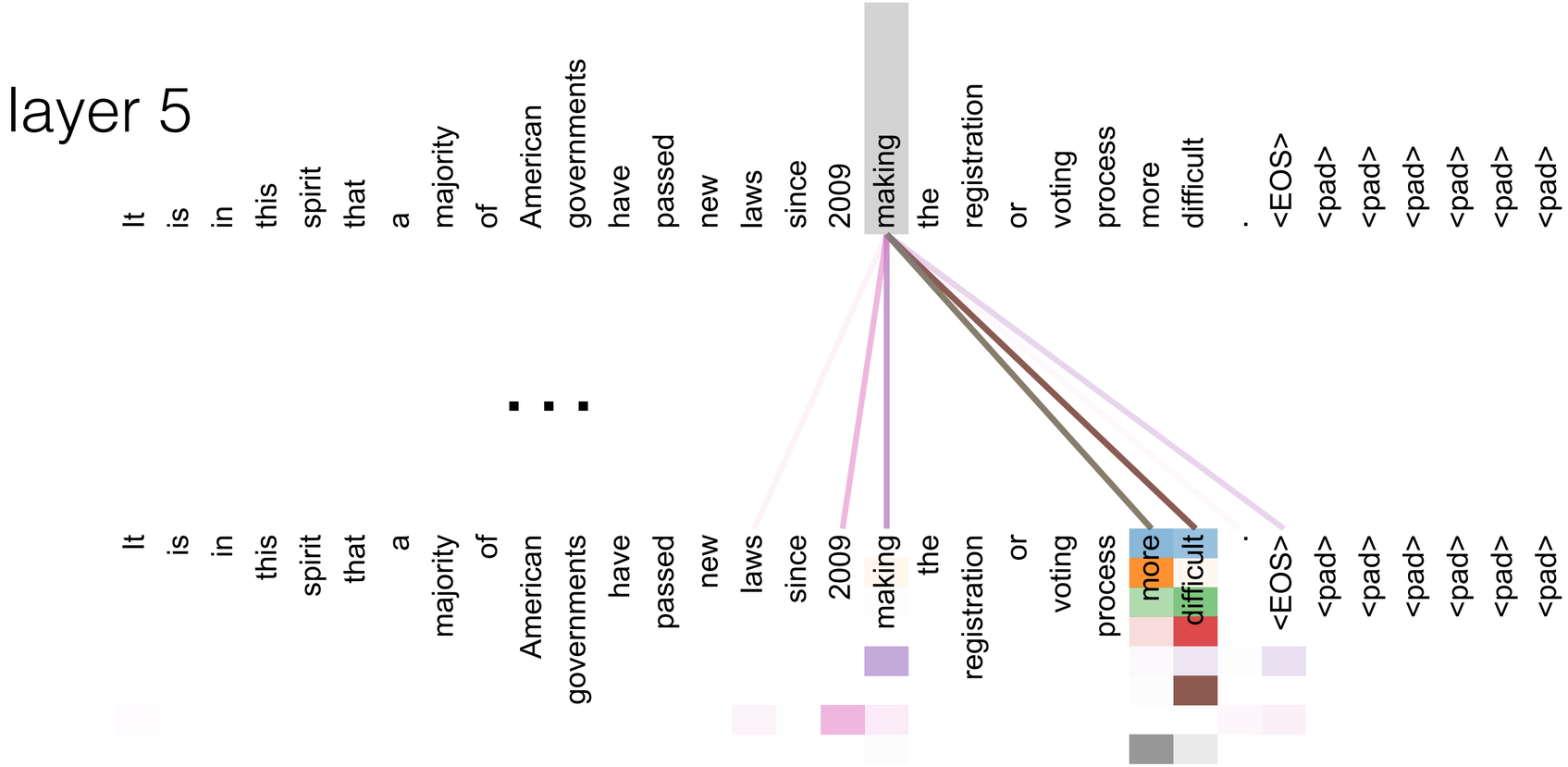
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- Many of the additional processing steps are not all necessary and could be removed/replaced [He et al.]

[Vaswani et al. 2017]

Multi-head attention: example

- Machine translation: higher attention layers can capture, e.g., phrase dependencies



[Vaswani et al. 2017]

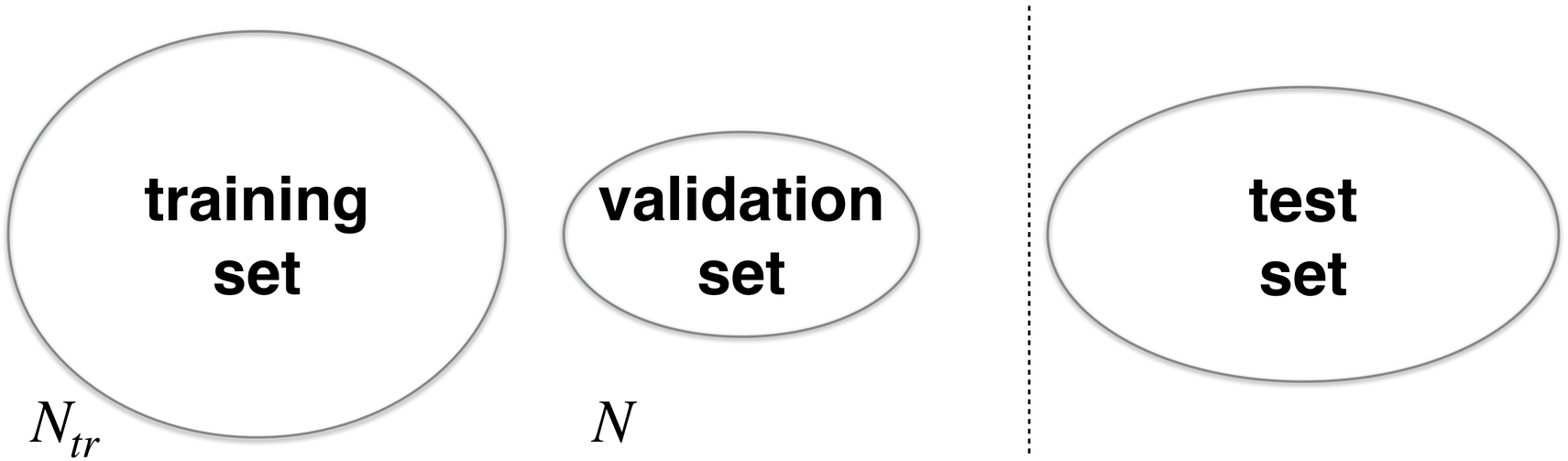
Outline

- Modeling sequences with state space models: Recurrent Neural Networks (**RNNs**)
- Modeling images (and related data): Convolutional Neural Networks (**CNNs**)
- Modeling graphs, data on graphs: Graph Neural Networks (**GNNs**)
- Modeling sequences, images, etc with **Transformers**

- Small steps towards generalization, uniform convergence, (algorithmic stability)

How to guarantee generalization?

- **training set:** used by the learning algorithm to fit the model parameters
- **validation set:** our proxy for test set, used for selecting modeling choices, including types of features (hyper-parameters that guide the learning algorithm)
- **test set:** only for deployment, final performance assessment; cannot be used iteratively to revise architectures etc



A typical procedure

- **training set:** used by the learning algorithm to fit adjustable parameters in the model, for each setting of the hyper-parameters, leading to data induced hypotheses $\hat{f}^{(l)},\ l = 1,\ldots,M$ that we need to select among

overall model + hyper-parameters (e.g., learning rate, features, architecture, optimization choices) **+** **training set** N_{tr} **=** $\{\hat{f}^{(l)},\ l = 1,\ldots,M\} = \hat{\mathscr{F}}_M$

- **validation set:** used to evaluate which one of $\hat{f}^{(l)},\ l = 1,\ldots,M$ we should adopt

induced hypotheses from training data $\hat{\mathscr{F}}_M$ **+** **validation set** N **=** $\hat{f} \in \hat{\mathscr{F}}_M$ selected predictor

Hyper-parameters, empirical risk, and generalization

- **Validation set:** $S = \{(x^i, y^i), i = 1, \ldots, N\}$, $(x^i, y^i) \sim P$ iid. E.g., $x^i \in \mathbb{R}^d, y^i \in \{-1, 1\}$
- **Test examples:** $(x, y) \sim P$ (same distribution)
- **Loss function:** $L(y, f(x))$ where $f(x)$ is our predictor, returning either a class label or a probability distribution over labels
- **Empirical validation risk:** $R_S(f) = \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i))$
- **Expected (test) risk:** $R(f) = E_{(x,y)\sim P} L(y, f(x))$
- **Hypothesis class:** $f \in \hat{\mathcal{F}}_M$ induced from candidate solutions based on training data (checkpoints, different hyper-parameter settings)
- We typically select $\hat{f} = \text{argmin}_{f \in \hat{\mathcal{F}}_M} R_S(f)$, hoping that $R_S(\hat{f}) \approx R(\hat{f})$

Empirical risk & generalization

- We are choosing among M hypotheses based on N iid examples
- Let's assume that the loss function is bounded within [0,1]. This can be always enforced on validation set regardless of the loss used for training

- We would like to ensure that the empirical validation risk $R_S(\hat{f})$ for the chosen classifier is close to the corresponding test risk $R(\hat{f})$ with high probability (so it's indeed valid to select the hypothesis with the best $R_S(\hat{f})$)
- We consider a slightly stronger requirement, that this holds uniformly for $f \in \hat{\mathcal{F}}_M$

$$P_{S\sim P^n}(\forall f \in \hat{\mathcal{F}}_M, |R_S(f) - R(f)| \leq \epsilon) \geq 1 - \delta$$

- The goal is to understand $\epsilon = \epsilon(\delta, N, M)$

Empirical risk & generalization

- We are choosing among M hypotheses based on N iid examples
- Let's assume that the loss function is bounded within [0,1]. This can be always enforced on validation set regardless of the loss used for training

- We would like to ensure that the empirical validation risk $R_S(\hat{f})$ for the chosen classifier is close to the corresponding test risk $R(\hat{f})$ with high probability (so it's indeed valid to select the hypothesis with the best $R_S(\hat{f})$)
- We consider a slightly stronger requirement, that this holds uniformly for $f \in \hat{\mathcal{F}}_M$

$$P_{S\sim P^n}(\forall f \in \hat{\mathcal{F}}_M, |R_S(f) - R(f)| \leq \epsilon) \geq 1 - \delta$$

- The goal is to understand $\epsilon = \epsilon(\delta, N, M)$

- We'll use an equivalent statement $P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M : \ |R_S(f) - R(f)| > \epsilon) < \delta$

Empirical risk & generalization

$$P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M : \ |R_S(f) - R(f)| > \epsilon) < \delta$$

- The statement ties together δ (prob that guarantee fails), ϵ (deviation from empirical), N (# of validation examples), and M (# of hypotheses we choose among)
- Implications for the smallest $\epsilon = \epsilon(\delta, N, M)$

$$N \uparrow \Rightarrow$$

Empirical risk & generalization

$$P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M : \ |R_S(f) - R(f)| > \epsilon) < \delta$$

- The statement ties together δ (prob that guarantee fails), ϵ (deviation from empirical), N (# of validation examples), and M (# of hypotheses we choose among)
- Implications for the smallest $\epsilon = \epsilon(\delta, N, M)$

$$N \uparrow \Rightarrow \epsilon \downarrow$$

$$\delta \uparrow \Rightarrow$$

Empirical risk & generalization

$$P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M : \ |R_S(f) - R(f)| > \epsilon) < \delta$$

- The statement ties together δ (prob that guarantee fails), ϵ (deviation from empirical), N (# of validation examples), and M (# of hypotheses we choose among)
- Implications for the smallest $\epsilon = \epsilon(\delta, N, M)$

$$N \uparrow \Rightarrow \epsilon \downarrow$$

$$\delta \uparrow \Rightarrow \epsilon \downarrow$$

$$M \uparrow \Rightarrow$$

Empirical risk & generalization

$$P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M : \ |R_S(f) - R(f)| > \epsilon) < \delta$$

- The statement ties together δ (prob that guarantee fails), ϵ (deviation from empirical), N (# of validation examples), and M (# of hypotheses we choose among)
- Implications for the smallest $\epsilon = \epsilon(\delta, N, M)$

$$N \uparrow \Rightarrow \epsilon \downarrow$$

$$\delta \uparrow \Rightarrow \epsilon \downarrow$$

$$M \uparrow \Rightarrow \epsilon \uparrow$$

Derivation (tightest ϵ)

- We derive here the smallest $\epsilon = \epsilon(\delta, N, M)$ that the statement allows.

$$P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M : \ |R_S(f) - R(f)| > \epsilon) < \delta$$

- We start with a basic Hoeffding's inequality for the average of independent bounded random variables (recall that we assumed that the loss is bounded within [0,1]): for a single classifier f

$$P_{S\sim P^n}(|R_S(f) - R(f)| > \epsilon) \leq 2\exp(-2N\epsilon^2)$$

Derivation (tightest ϵ)

- Hoeffding's inequality for a single classifier

$$P_{S\sim P^n}(|R_S(f) - R(f)| > \epsilon) \leq 2\exp(-2N\epsilon^2)$$

- We can then use the union bound $P(A_1 \cup A_2 \cup \ldots \cup A_M) \leq \sum_{j=1}^{M} P(A_j)$ for events $A_j = \{S: \ |R_S(f^j) - R(f^j)| > \epsilon\}$

$$P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M: \ |R_S(f) - R(f)| > \epsilon)$$

Derivation (tightest ϵ)

- Hoeffding's inequality for a single classifier

$$P_{S\sim P^n}(\,|R_S(f) - R(f)\,| > \epsilon) \leq 2\exp(-2N\epsilon^2)$$

- We can then use the union bound $P(A_1 \cup A_2 \cup \ldots \cup A_M) \leq \sum_{j=1}^{M} P(A_j)$ for events $A_j = \{S:\ |R_S(f^j) - R(f^j)| > \epsilon\}$

$$P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M:\ |R_S(f) - R(f)| > \epsilon) \leq \sum_{f\in\hat{\mathcal{F}}_M} P_{S\sim P^n}(\,|R_S(f) - R(f)\,| > \epsilon)$$

Derivation (tightest ϵ)

- Hoeffding's inequality for a single classifier

$$P_{S\sim P^n}(|R_S(f) - R(f)| > \epsilon) \leq 2\exp(-2N\epsilon^2)$$

- We can then use the union bound $P(A_1 \cup A_2 \cup \ldots \cup A_M) \leq \sum_{j=1}^{M} P(A_j)$ for events $A_j = \{S: \ |R_S(f^j) - R(f^j)| > \epsilon\}$

$$P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M: \ |R_S(f) - R(f)| > \epsilon) \leq \sum_{f\in\hat{\mathcal{F}}_M} P_{S\sim P^n}(|R_S(f) - R(f)| > \epsilon)$$

$$\leq \sum_{f\in\hat{\mathcal{F}}_M} 2\exp(-2N\epsilon^2)$$

Derivation (tightest ϵ)

- Hoeffding's inequality for a single classifier

$$P_{S\sim P^n}(\,|R_S(f) - R(f)\,| > \epsilon) \leq 2\exp(-2N\epsilon^2)$$

- We can then use the union bound $P(A_1 \cup A_2 \cup \ldots \cup A_M) \leq \sum_{j=1}^{M} P(A_j)$ for events $A_j = \{S:\ |R_S(f^j) - R(f^j)| > \epsilon\}$

$$\begin{aligned} P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M:\ |R_S(f) - R(f)| > \epsilon) &\leq \sum_{f\in\hat{\mathcal{F}}_M} P_{S\sim P^n}(\,|R_S(f) - R(f)\,| > \epsilon) \\ &\leq \sum_{f\in\hat{\mathcal{F}}_M} 2\exp(-2N\epsilon^2) \\ &= 2M\exp(-2N\epsilon^2) \equiv \delta \end{aligned}$$

Derivation (tightest ϵ)

- Hoeffding's inequality for a single classifier

$$P_{S\sim P^n}(\,|R_S(f) - R(f)\,| > \epsilon) \leq 2\exp(-2N\epsilon^2)$$

- We can then use the union bound $P(A_1 \cup A_2 \cup \ldots \cup A_M) \leq \sum_{j=1}^{M} P(A_j)$ for events $A_j = \{S:\ |R_S(f^j) - R(f^j)| > \epsilon\}$

$$\begin{aligned} P_{S\sim P^n}(\exists f \in \hat{\mathcal{F}}_M:\ |R_S(f) - R(f)| > \epsilon) &\leq \sum_{f\in\hat{\mathcal{F}}_M} P_{S\sim P^n}(\,|R_S(f) - R(f)\,| > \epsilon) \\ &\leq \sum_{f\in\hat{\mathcal{F}}_M} 2\exp(-2N\epsilon^2) \\ &= 2M\exp(-2N\epsilon^2) \equiv \delta \end{aligned}$$

$$\Rightarrow \epsilon(\delta, N, M) = \sqrt{\frac{\log(2M/\delta)}{2N}}$$

Is the result relevant?

- Suppose we do a lot of hyper-parameter optimizations, essentially discretizing continuous choices (including when to stop training).
- Taken together, we wish to explore $M = 1000{,}000$ different hyper-parameter choices
- We would like the result to hold with 0.95 probability ($\delta = 0.05$ failure rate)
- We have $N = 4000$ validation examples

$$\Rightarrow \epsilon(\delta, N, M) = \sqrt{\frac{\log(2M/\delta)}{2N}} = 0.046$$

- In other words, with probability at least 0.95 (over the choice of validation data), the true error of our chosen best predictor deviates at most 0.05 from its measured validation error

References

- Bishop & Bishop, “Deep Learning”, chapter 12
- Vaswani et al., “Attention is all you need”, https://arxiv.org/abs/1706.03762, 2017
- He et al., “Simplifying Transformer Blocks”, https://arxiv.org/abs/2311.01906
- Hardt et al. “Train faster, generalize better: Stability of stochastic gradient descent”, https://proceedings.mlr.press/v48/hardt16.pdf
- Shalev-Shwartz et al., “Understanding Machine Learning: From Theory to Algorithms”, chapters 3 and 4