
بسم الله الرحمن الرحیم

یادگیری عمیق

مدرس: محمدرضا محمدی

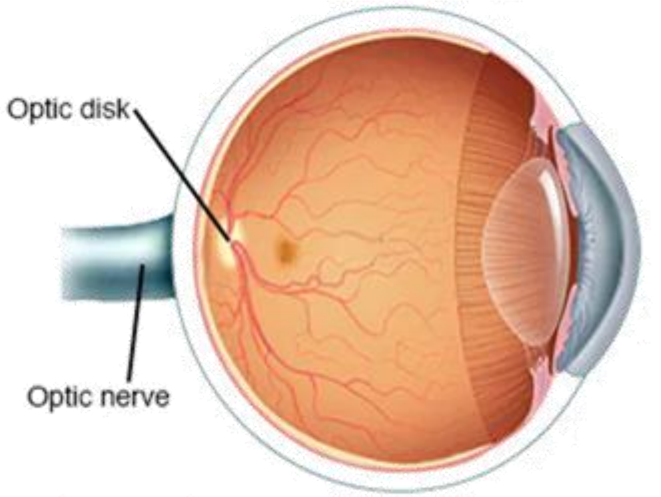
بهار ۱۴۰۲

مکانیزم‌های توجه

Attention Mechanisms

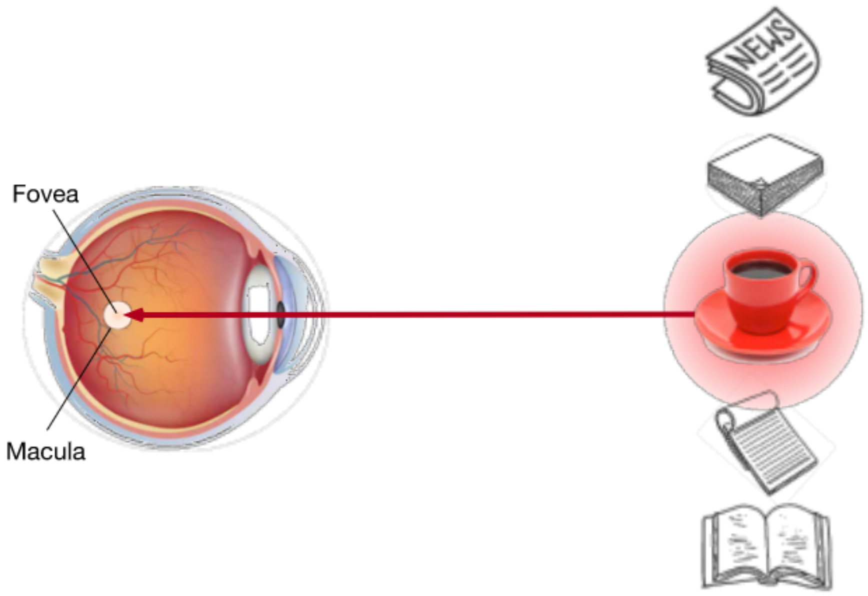
مکانیزم‌های توجه

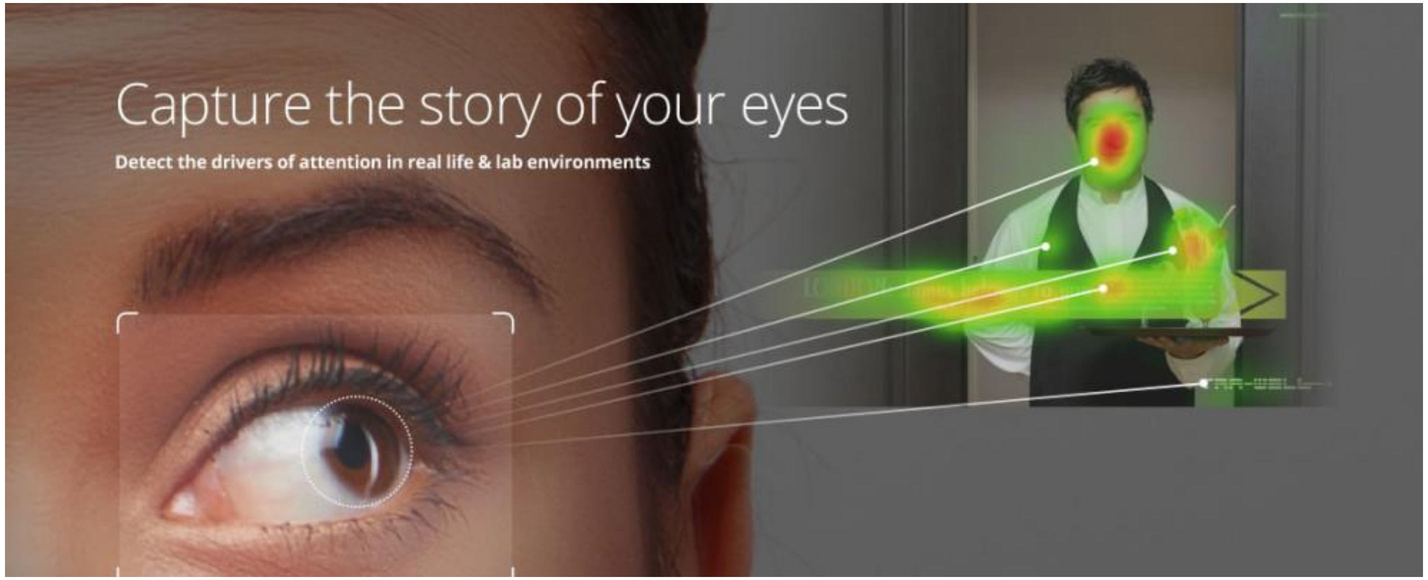
- عصب بینایی ورودی عظیمی را دریافت می‌کند که بسیار فراتر از آن چیزی است که مغز قادر به پردازش کامل آن باشد
- با این حال، مغز می‌تواند به اشیاء مهم توجه نماید
- توانایی توجه به بخش کوچکی از اطلاعات دارای اهمیت تکاملی بسیار زیادی است



نشانه‌های توجه در زیست‌شناسی

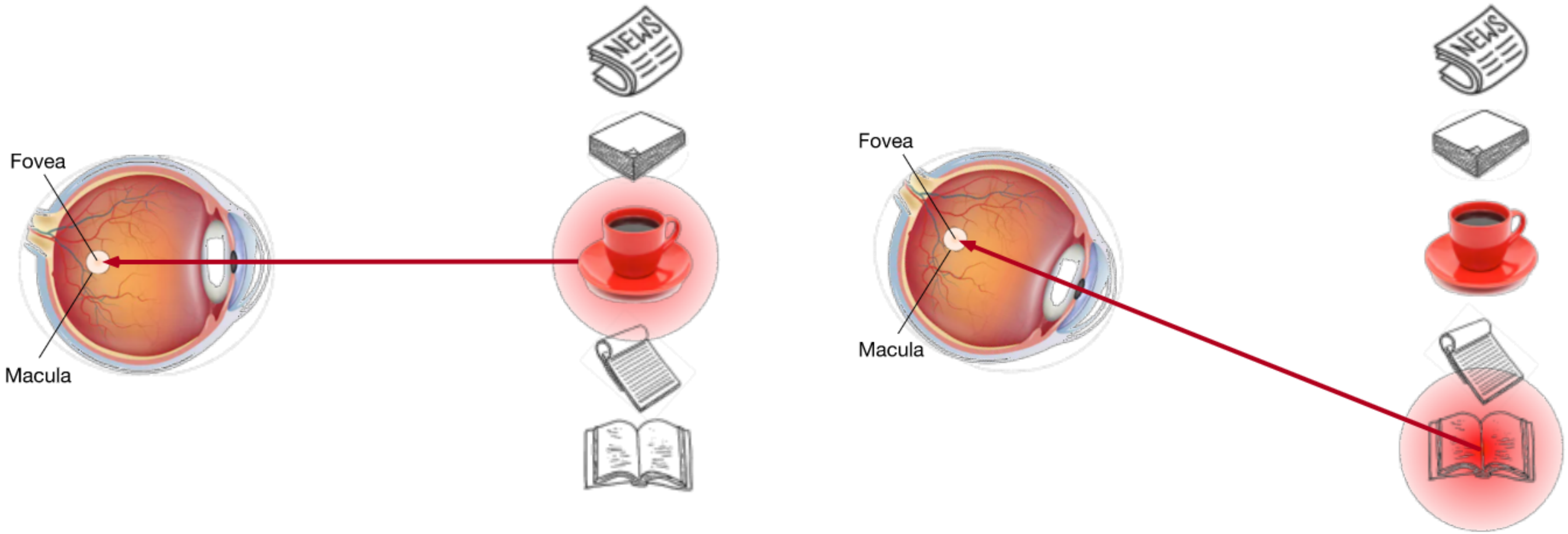
- حالت غیرارادی (nonvolitional) مبتنی بر برجستگی (saliency) اشیاء در محیط است
  - به عنوان مثال، یک فنجان قهوه قرمز در میان روزنامه، مقاله، دفترچه و کتاب (همگی غیررنگی) توجه را جلب می‌کند





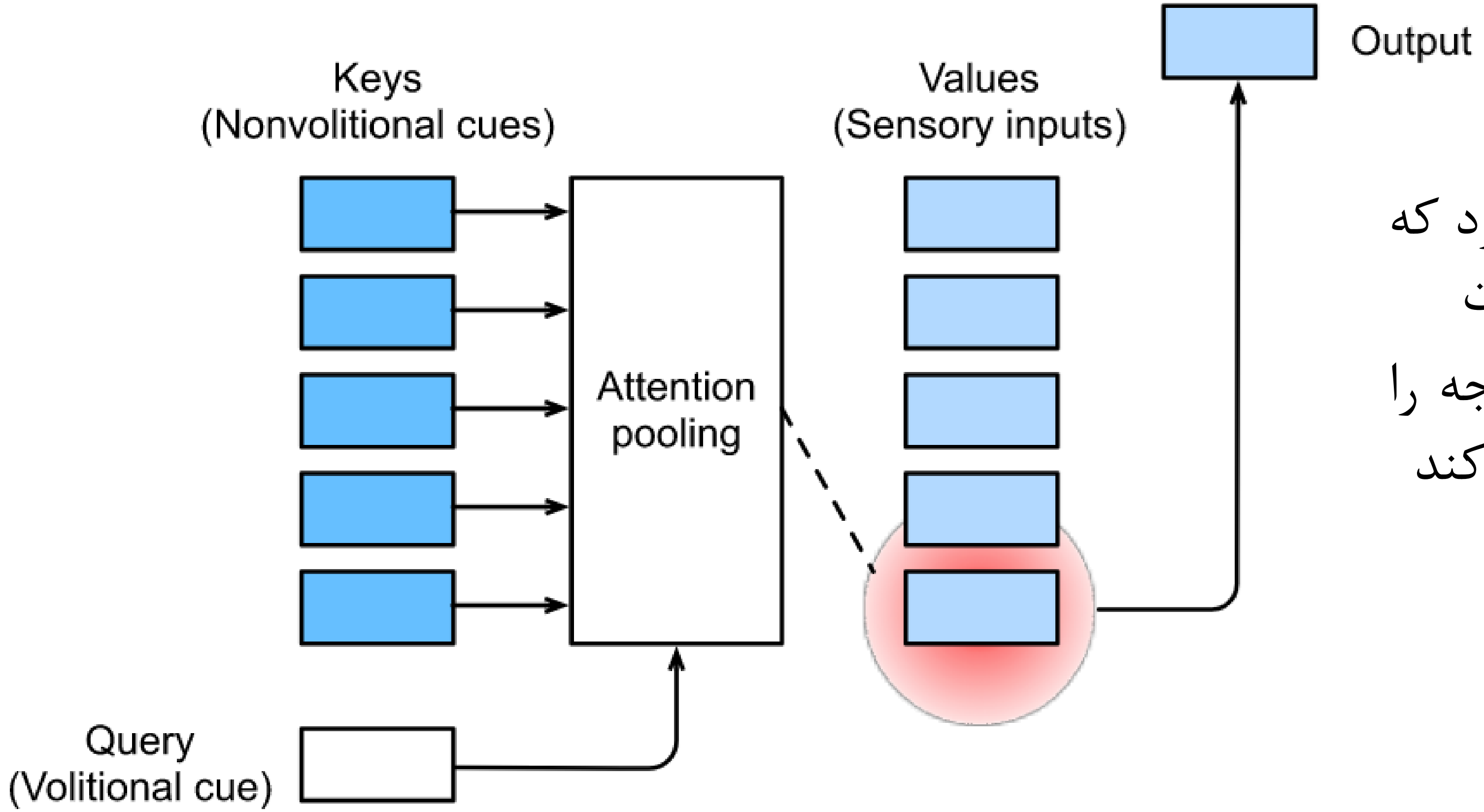
نشانه‌های توجه در زیست‌شناسی

- حالت غیرارادی (nonvolitional) مبتنی بر برجستگی (saliency) اشیاء در محیط است
- حالت ارادی (volitional) وابسته به وظیفه (task-dependent) است
  - پس از نوشیدن قهوه ممکن است بخواهید کتاب بخوانید و توجه شما به کتاب جلب خواهد شد



مکانیزم‌های توجه

- با الهام از نشانه‌های توجه غیرارادی و ارادی، در ادامه چارچوبی برای طراحی مکانیزم‌های توجه با ترکیب این دو نشانه شرح داده می‌شود



  - Values: مقادیر ورودی
  - Keys: به ازای هر value یک key وجود دارد که متناظر با نشانه غیرارادی مربوط به آن است
  - Query: متناظر با نشانه ارادی است و توجه را متناسب با وظیفه مورد نظر جهت‌دهی می‌کند

ادغام توجه

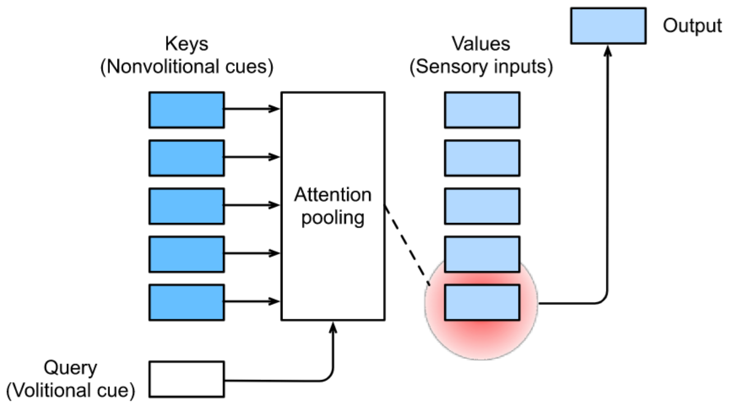
- برای بررسی دقیق‌تر مکانیزم توجه، در ادامه مثال ساده Nadaraya-Watson kernel regression را بررسی می‌کنیم
- به عنوان مجموعه داده، ۵۰ نمونه آموزشی و ۵۰ نمونه آزمون از رابطه زیر تولید می‌کنیم:
  - ϵ یک متغیر تصادفی نرمال با میانگین ۰ و انحراف معیار ۰.۵ است

$$y_i = 2\sin(x_i) + x_i^{0.8} + \epsilon$$

```
n_train = 50
x_train, _ = torch.sort(torch.rand(n_train) * 5)

def f(x):
    return 2 * torch.sin(x) + x**0.8

y_train = f(x_train) + torch.normal(0.0, 0.5, (n_train,))
x_test = torch.arange(0, 5, 0.1)
y_truth = f(x_test)
```

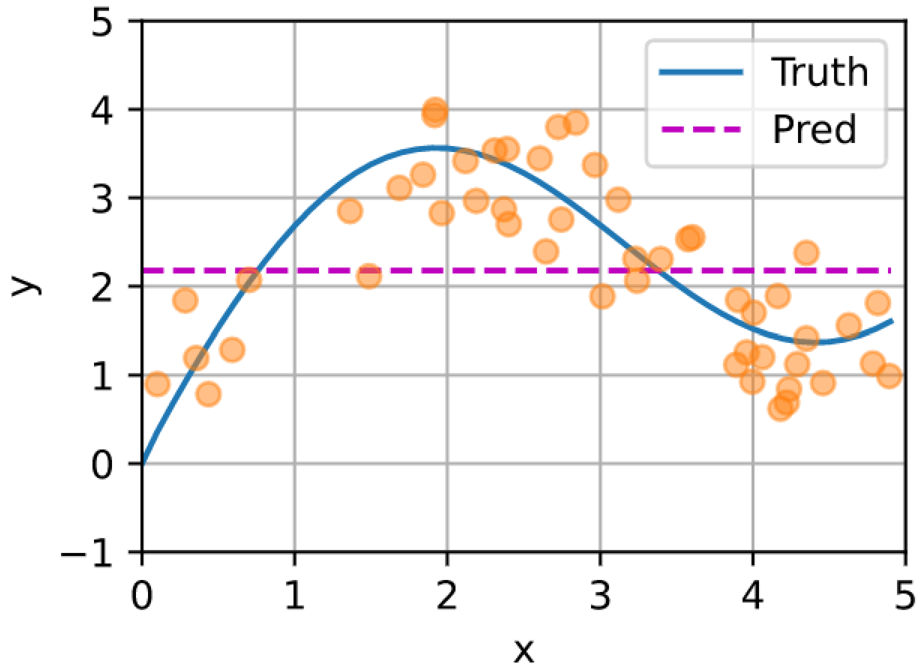


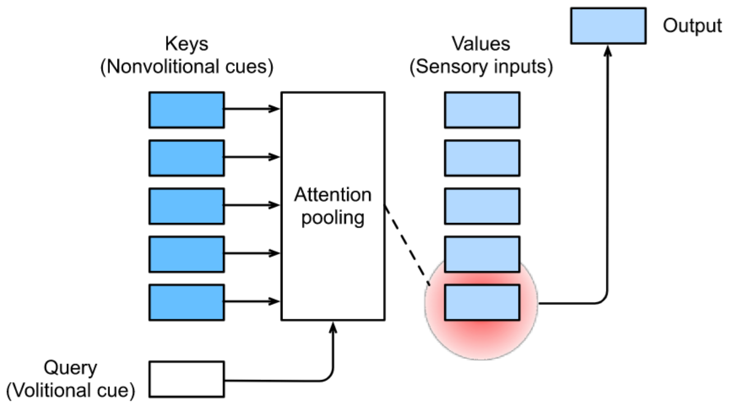
ادغام میانگین

$$y_i = 2\sin(x_i) + x_i^{0.8} + \epsilon$$

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$$

- برای شروع از ساده‌ترین برآوردگر استفاده می‌کنیم: محاسبه میانگین!
  - در این روش برای تخمین هیچ توجهی به x نمی‌شود





ادغام توجه غیرپارامتری

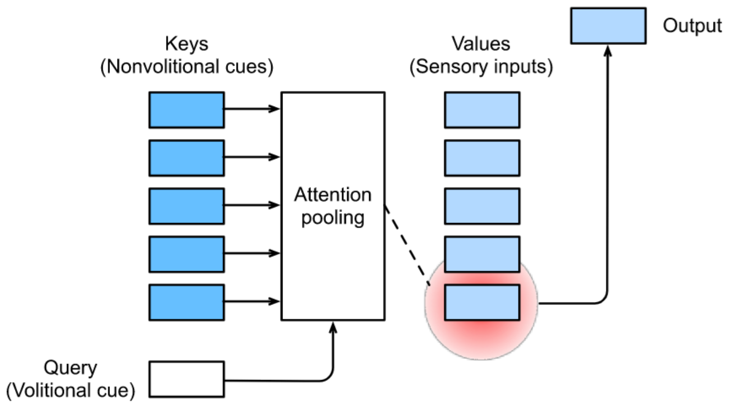
$$y_i = 2\sin(x_i) + x_i^{0.8} + \epsilon$$

- ایده مطرح شده توسط Nadaraya و Watson این است که در محاسبه خروجی هر y_i متناسب با مقدار x_i وزن بگیرد
  - که K یک هسته (kernel) است

$$f(x) = \sum_{i=1}^{n} \frac{K(x - x_i)}{\sum_{j=1}^{n} K(x - x_j)} y_i$$

- می‌توان این معادله را با نگاه مکانیزم توجه بازنویسی کرد
  - که x متناظر با query و (x_i, y_i) متناظر با جفت key-value است
  - وزن توجه $\alpha(x, x_i)$ برای هر y_i بر اساس مقادیر query و key محاسبه می‌شود
  - برای هر query، وزن‌های توجه باید از یک توزیع احتمال معتبر باشند
    - نامنفی باشند و مجموع آنها ۱ شود
    - هسته گاوسی یک انتخاب مناسب است

$$f(x) = \sum_{i=1}^{n} \alpha(x, x_i) y_i$$



ادغام توجه غیرپارامتری

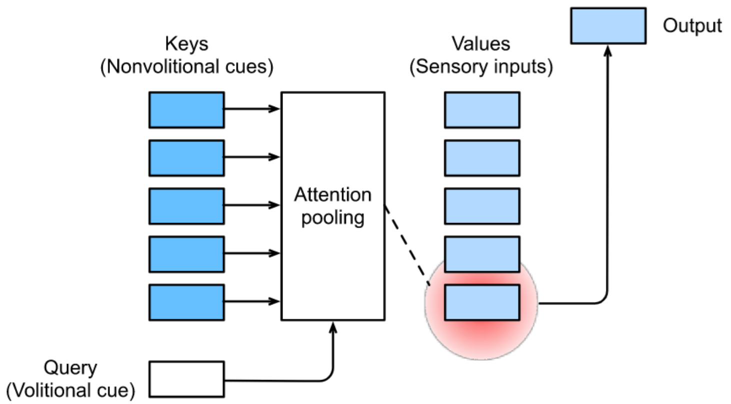
$$y_i = 2\sin(x_i) + x_i^{0.8} + \epsilon$$

$$f(x) = \sum_{i=1}^{n} \alpha(x, x_i) y_i$$

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$$

$$\Rightarrow f(x) = \sum_{i=1}^{n} \frac{\exp\left(-\frac{1}{2}(x - x_i)^2\right)}{\sum_{j=1}^{n} \exp\left(-\frac{1}{2}(x - x_j)^2\right)} y_i$$

$$= \sum_{i=1}^{n} \operatorname{softmax}\left(-\frac{1}{2}(x - x_i)^2\right) y_i$$



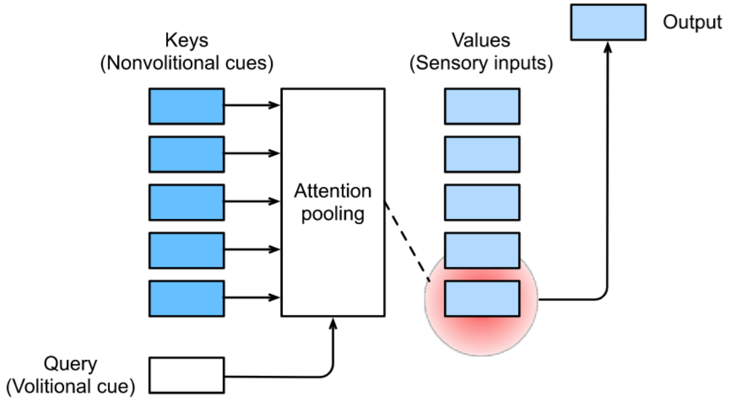
- هر x_i (key) که به x (query) نزدیکتر باشد توجه بیشتری را به خوب جلب خواهد کرد و y_i (value) مربوط به آن وزن بزرگتری خواهد داشت
- این مثال نمونه‌ای از ادغام توجه غیرپارامتری است

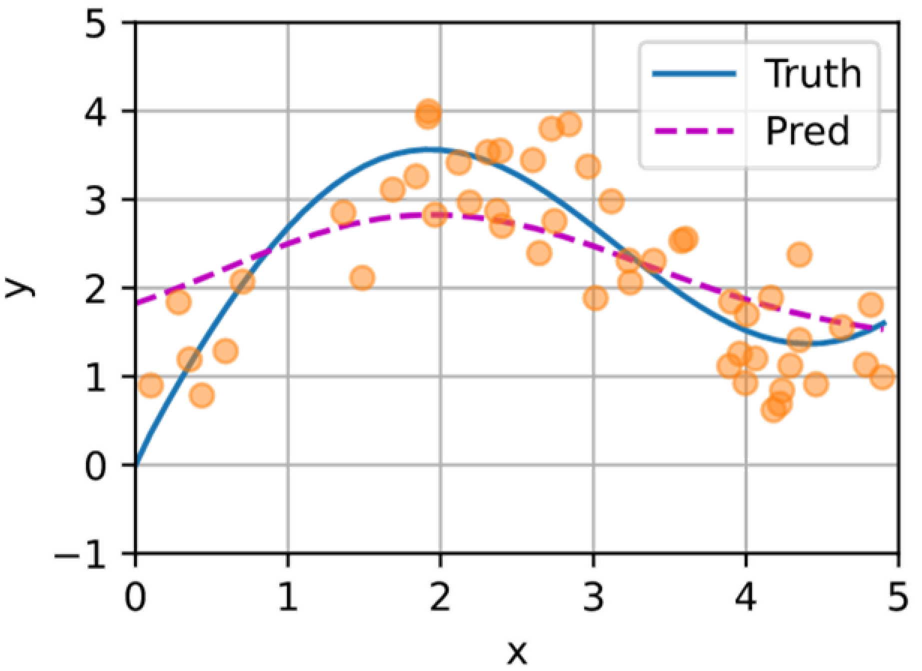
ادغام توجه غیرپارامتری

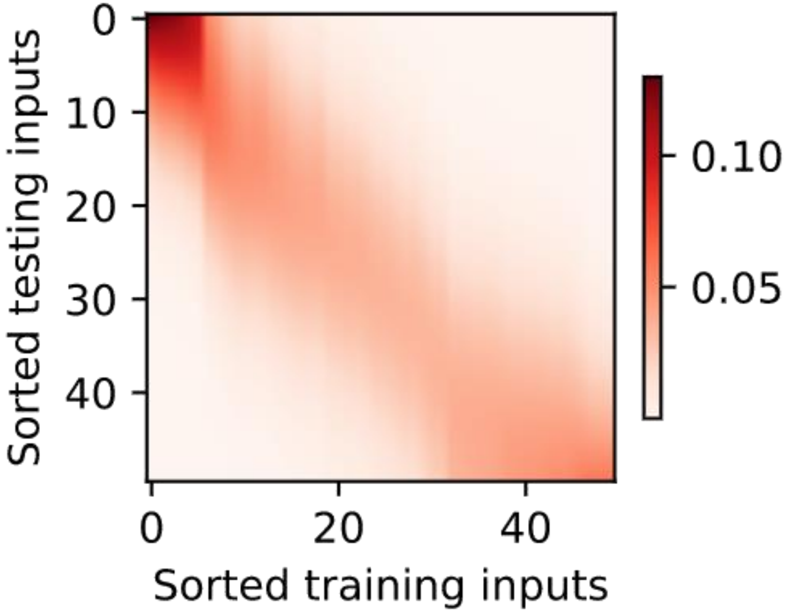
$$y_i = 2\sin(x_i) + x_i^{0.8} + \epsilon$$

$$f(x) = \sum_{i=1}^{n} \mathrm{softmax}\left(-\frac{1}{2}(x - x_i)^2\right) y_i$$

- می‌توان برای تحلیل بهتر، وزن‌های توجه را نمایش داد
  - جفت query-keyهای نزدیکتر به هم دارای وزن‌های بزرگتری هستند







ادغام توجه پارامتری

- به سادگی می‌توان پارامترهای قابل آموزش به ادغام توجه اضافه کرد
  - به عنوان مثال، می‌توان انحراف معیار هسته گاوسی را آموزش داد
  - این پارامتر با روش‌های مبتنی بر گرادیان قابل آموزش است

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(wu)^2}{2}\right)$$

$$f(x) = \sum_{i=1}^{n} \frac{\exp\left(-\frac{1}{2}\left(w(x - x_i)\right)^2\right)}{\sum_{j=1}^{n} \exp\left(-\frac{1}{2}\left(w(x - x_j)\right)^2\right)} y_i$$

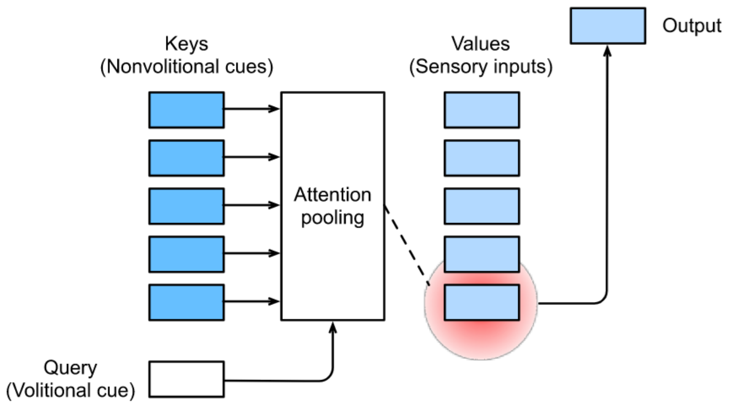
$$= \sum_{i=1}^{n} \mathrm{softmax}\left(-\frac{1}{2}\left(w(x - x_i)\right)^2\right) y_i$$

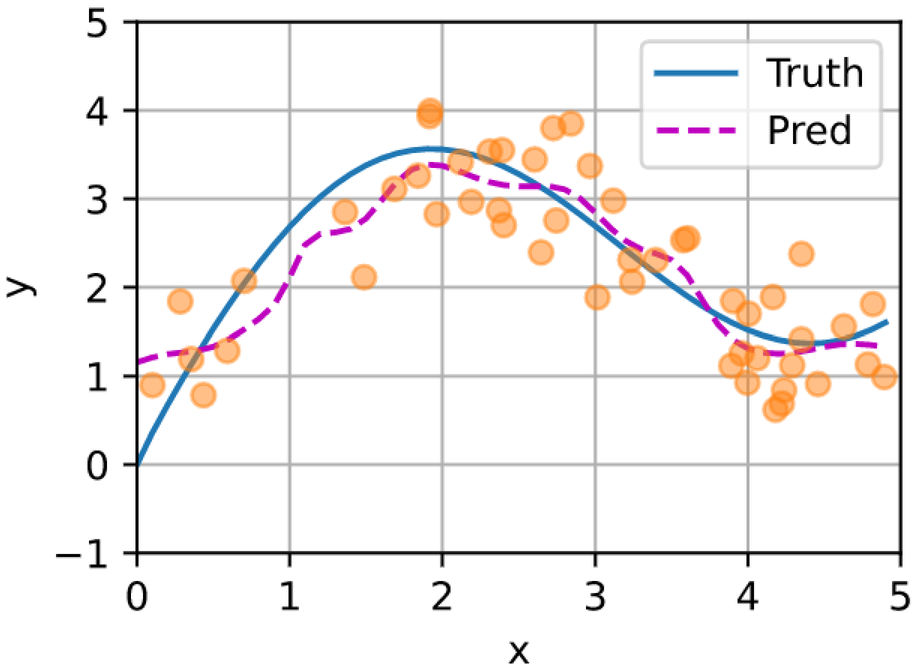
ادغام توجه پارامتری

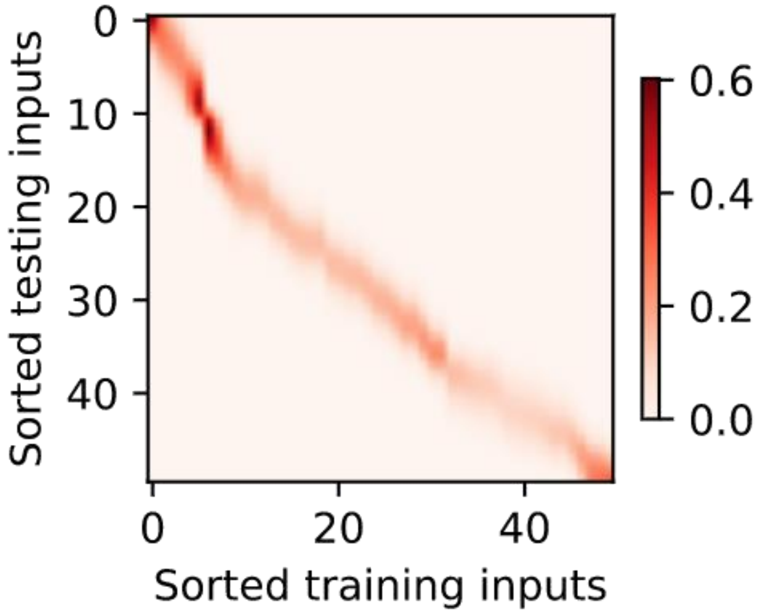
$$y_i = 2\sin(x_i) + x_i^{0.8} + \epsilon$$

$$f(x) = \sum_{i=1}^{n} \mathrm{softmax}\left(-\frac{1}{2}\left(w(x - x_i)\right)^2\right) y_i$$

- در این مثال نزدیک بودن اهمیت بیشتری داشته است و $w > 1$ شده است

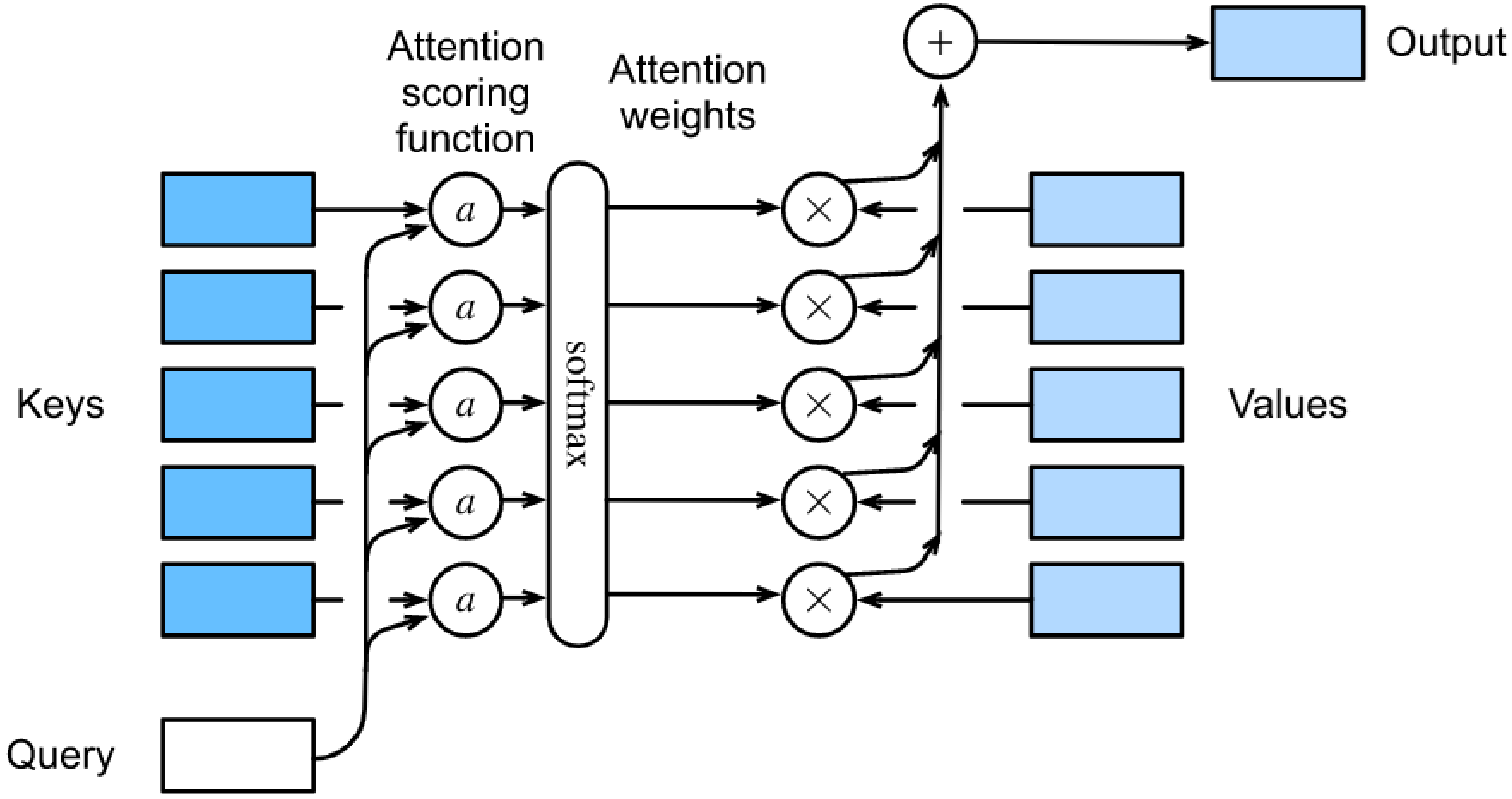






توابع امتیازدهی توجه

$$f(x) = \sum_{i=1}^{n} \text{softmax}\left(-\frac{1}{2}\left(w(x - x_i)\right)^2\right) y_i$$



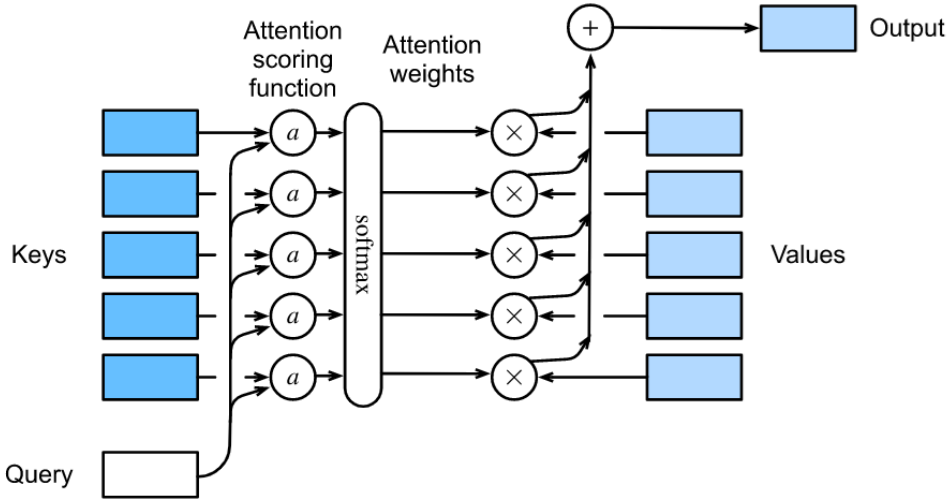
- در مثال قبل از یک هسته گاوسی استفاده کردیم و بخش توان آن به نوعی تابع امتیازدهی توجه (attention scoring function) است که وارد یک تابع softmax می‌شود و خروجی برابر با جمع وزن‌دار values با این ضرایب است

توابع امتیازدهی توجه

- فرض کنید $\mathbf{q} \in \mathbb{R}^q$ یک query و $(\mathbf{k}_1,\mathbf{v}_1),\ldots,(\mathbf{k}_m,\mathbf{v}_m)$ نشان‌دهنده m جفت key-value باشند که $\mathbf{k}_i \in \mathbb{R}^k$ و $\mathbf{v}_i \in \mathbb{R}^v$

$$f\big(\mathbf{q},(\mathbf{k}_1,\mathbf{v}_1),\ldots,(\mathbf{k}_m,\mathbf{v}_m)\big) = \sum_{i=1}^{n} \alpha(\mathbf{q},\mathbf{k}_i)\mathbf{v}_i$$

$$\alpha(\mathbf{q},\mathbf{k}_i) = \mathrm{softmax}\big(a(\mathbf{q},\mathbf{k}_i)\big) = \frac{\exp\big(a(\mathbf{q},\mathbf{k}_i)\big)}{\sum_{j=1}^{m}\exp\big(a(\mathbf{q},\mathbf{k}_j)\big)} \in \mathbb{R}$$



- انتخاب‌های مختلفی برای تابع امتیازدهی a می‌توان داشت

عملیات Softmax ماسک‌شده

- در بخش قبل با استفاده از تابع softmax یک توزیع احتمال به عنوان وزن‌های توجه ایجاد شد
- در برخی موارد، تمام مقادیر نباید به تابع ادغام توجه وارد شوند
  - به عنوان مثال، ممکن است به انتهای یک متن توکن pad اضافه کرده باشیم
  - برای آنکه توجه فقط روی توکن‌های معنی‌دار جلب شود، می‌توانیم یک طول دنباله معتبر تعیین کنیم تا در هنگام محاسبه softmax، مواردی که فراتر از این محدوده هستند را فیلتر کنیم
  - در پیاده‌سازی، $a(\mathbf{q},\mathbf{k}_i)$ تنها برای بخشی از iهای مجاور محاسبه می‌شود و سپس وارد تابع softmax می‌شوند و وزن سایر مقادیر صفر خواهد بود
  - در این مثال رنگ زرد مشخص‌کننده توکن‌های pad است که نباید به آنها توجه شود

Attention weights before softmax

3.8	4.4	3.0	3.6	3.2	3.2
3.4	3.6	3.8	4.0	3.8	4.4
3.4	4.8	3.0	4.4	3.8	4.2
3.2	3.4	4.6	5.0	3.6	4.4

Attention weights after softmax

0.24	0.45	0.11	0.2	0	0
1	0	0	0	0	0
0.082	0.33	0.055	0.22	0.12	0.18
0.16	0.19	0.65	0	0	0

Masking attention weights in PyTorch (juditacs.github.io)

توجه افزودنی

- ابعاد query و key ممکن است متفاوت باشد
  - در این موارد می‌توانیم از توجه افزودنی به عنوان تابع امتیازدهی استفاده کنیم

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

  - که $\mathbf{W}_q \in \mathbb{R}^{h\times q}$ و $\mathbf{W}_k \in \mathbb{R}^{h\times k}$ و $\mathbf{w}_v \in \mathbb{R}^{h}$ پارامترهای قابل آموزش هستند
  - توسط یک MLP دولایه که ورودی آن نسخه الحاق شده $\mathbf{q}$ و $\mathbf{k}$ باشد قابل پیاده‌سازی است که h ابرپارامتر آن است

توجه ضرب داخلی مقیاس‌شده

- یک طراحی کارآمدتر از لحاظ محاسباتی برای تابع امتیازدهی می‌تواند با استفاده از ضرب داخلی انجام شود

$$a(\mathbf{q},\mathbf{k}) = \mathbf{q}^{\mathrm{T}}\mathbf{k}$$

  - با این حال، ضرب داخلی مستلزم آن است که query و key دارای طول یکسان باشند

- اگر فرض کنید تمام عناصر query و key دارای میانگین صفر و واریانس ۱ باشند، حاصل $\mathbf{q}^{\mathrm{T}}\mathbf{k}$ دارای میانگین صفر و واریانس d خواهد بود

- برای آنکه واریانس حاصل ۱ بماند، می‌توانیم نتیجه ضرب داخلی را مقیاس کنیم

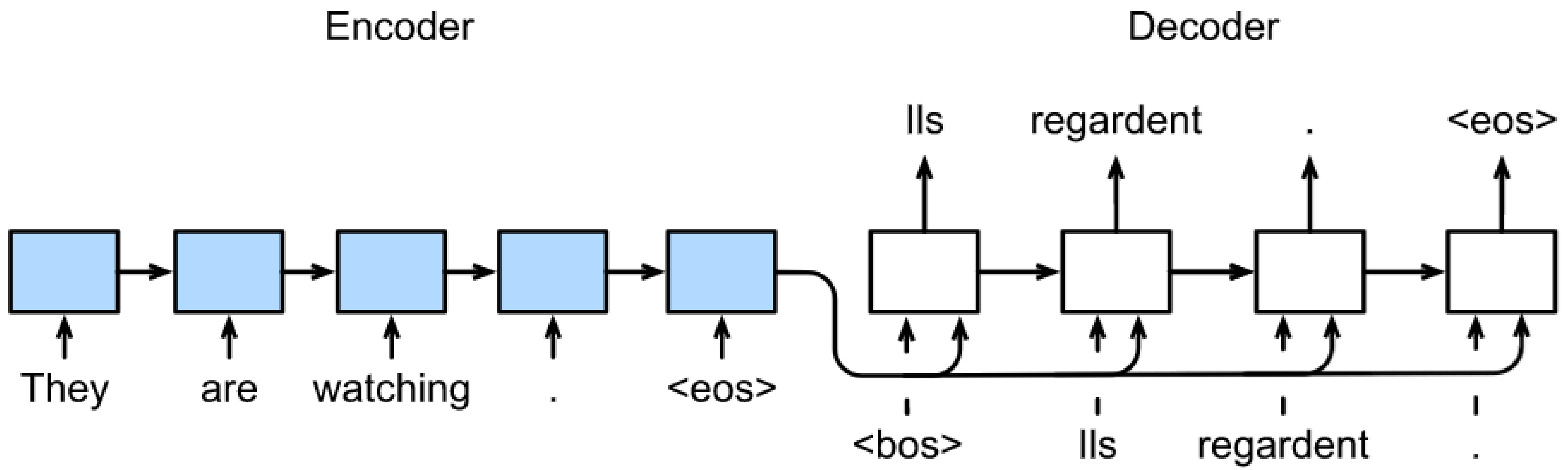
$$a(\mathbf{q},\mathbf{k}) = \mathbf{q}^{\mathrm{T}}\mathbf{k}/\sqrt{d}$$

- برای n بردار query می‌توان محاسبات را به صورت ماتریسی انجام داد:

$$\mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d}}\right)\mathbf{V} \in \mathbb{R}^{n\times v} \quad , \ \mathbf{Q} \in \mathbb{R}^{n\times d} \quad , \ \mathbf{K} \in \mathbb{R}^{m\times d} \quad , \ \mathbf{V} \in \mathbb{R}^{m\times v}$$

توجه Bahdanau

- در فصل قبل مسئله ترجمه ماشینی با استفاده از معماری encoder-decoder را بررسی کردیم
  - شبکه بازگشتی encoder یک دنباله با طول متغیر را به یک متغیر مفهومی (context) با ابعاد ثابت تبدیل می‌کند (**c**)
  - شبکه بازگشتی decoder یک دنباله با طول متغیر را بر اساس متغیر مفهومی تولید می‌کند
    - در این مثال، متغیر **c** یکسان در تمام گام‌ها استفاده می‌شود درحالیکه همه ورودی برای تولید یک توکن نیاز نیست

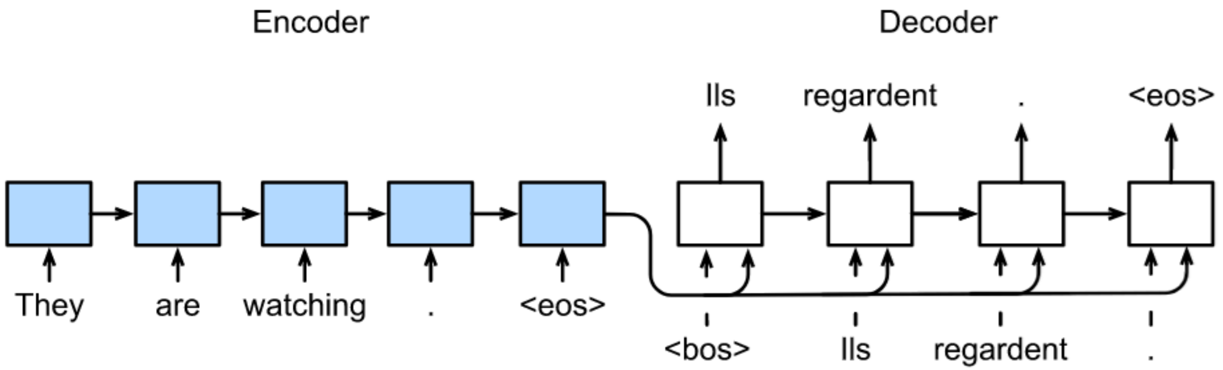


توجه Bahdanau

- در مدل پیشنهادی، بجای استفاده از $\mathbf{c}$ یکسان در تمام گام‌ها، از $\mathbf{c}_{t'}$ استفاده می‌کنیم
  - فرض کنید دنباله ورودی دارای T توکن باشد، $\mathbf{c}_{t'}$ به صورت زیر محاسبه می‌شود:
  - که $\mathbf{s}_{t'-1}$ حالت پنهان decoder در گام $t'-1$ به عنوان query است
  - و $\mathbf{h}_t$ حالت پنهان encoder در گام t به عنوان key و همچنین value است
  - با توجه به اینکه طول $\mathbf{s}$ و $\mathbf{h}$ می‌تواند متفاوت باشد، از تابع امتیازدهی توجه افزودنی استفاده می‌کنیم

$$\mathbf{c}_{t'} = \sum_{t=1}^{T} \alpha(\mathbf{s}_{t'-1}, \mathbf{h}_t)\mathbf{h}_t$$

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^{\mathrm{T}} \tanh\left(\mathbf{W}_q \mathbf{q} + \mathbf{W}_k \mathbf{k}\right)$$



Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint,* 2014.

Bahdanau توجه

$$\mathbf{c}_{t'} = \sum_{t=1}^{T} \alpha(\mathbf{s}_{t'-1}, \mathbf{h}_t)\mathbf{h}_t$$

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^{\mathrm{T}} \tanh(\mathbf{W}_q\mathbf{q} + \mathbf{W}_k\mathbf{k})$$