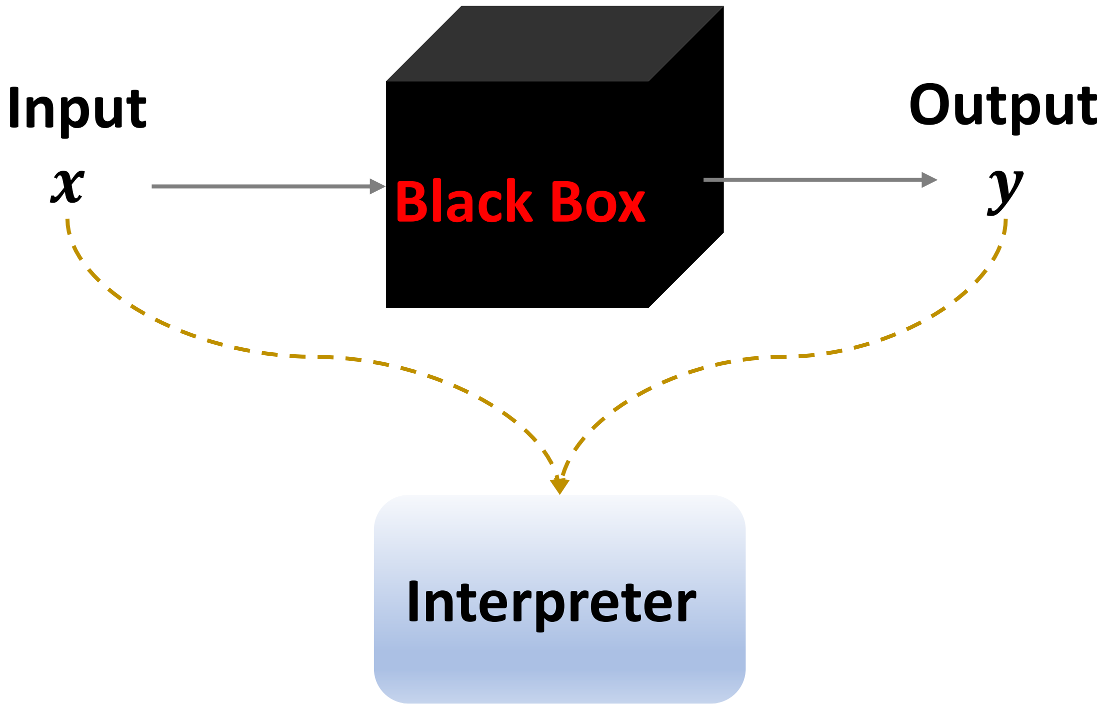
CS 4501/6501 Interpretable Machine Learning

Post-hoc explanations: gradient/attention-based methods

Hanjie Chen, Yangfeng Ji
Department of Computer Science
University of Virginia
{hc9mx, yangfeng}@virginia.edu

Explaining Black-box Model

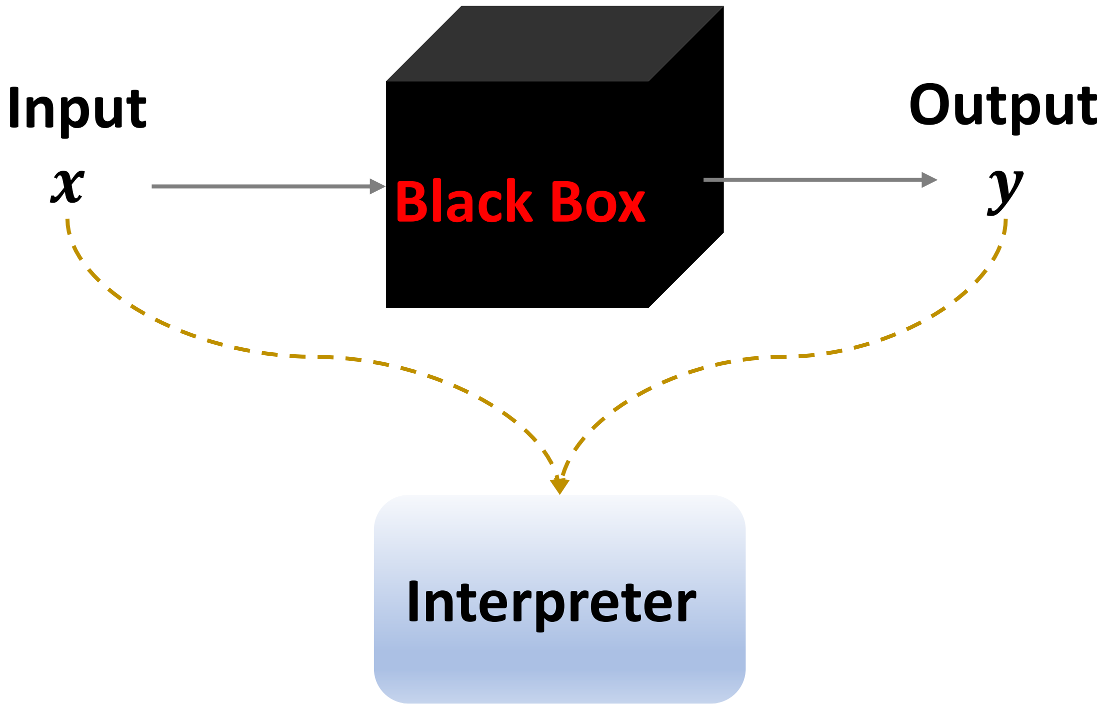
Perturbation-based methods



- Model-agnostic (black-box)
- Perturbing the input and observing model prediction change
- Extracting relationships between input features and the output

Explaining Black-box Model

Perturbation-based methods

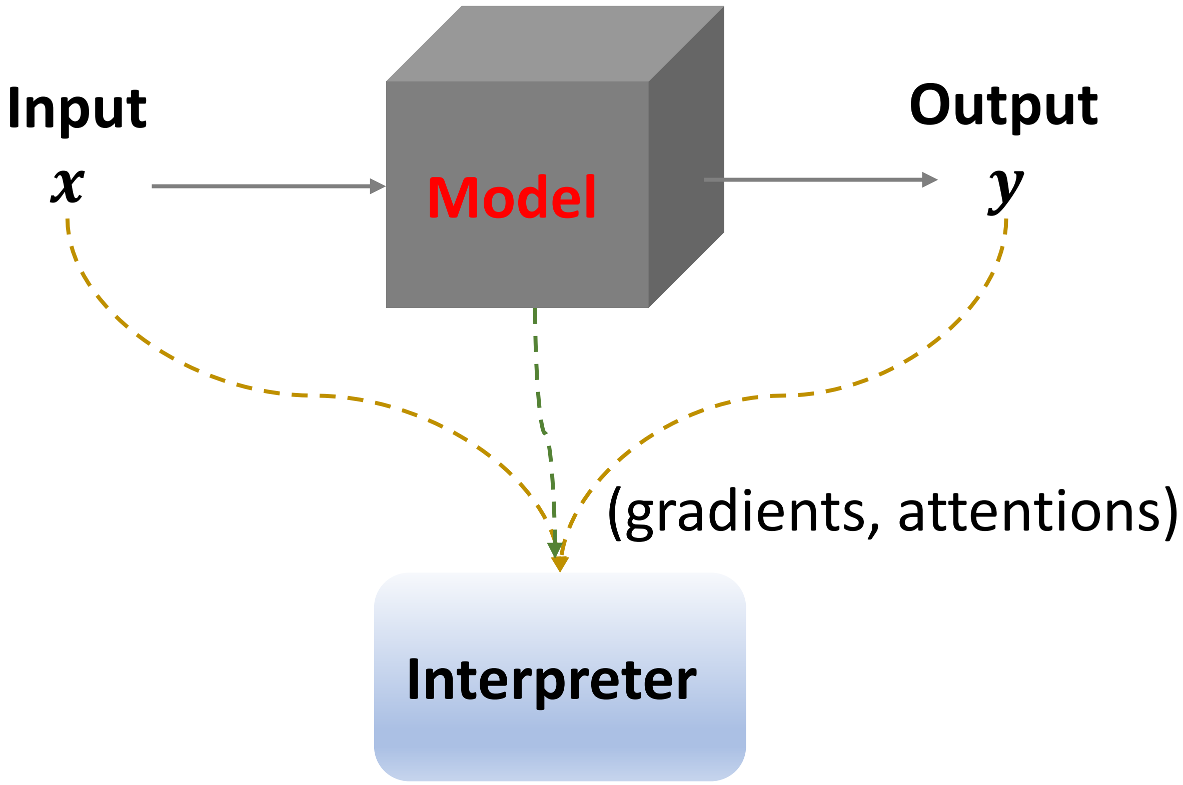


- Model-agnostic (black-box)
- Perturbing the input and observing model prediction change
- Extracting relationships between input features and the output

- Applicable to any black-box models
- Computational complexity

Explaining Black-box Model

Additional information from the model



- Model-dependent (white-box)
- Additional information: gradients, attentions
- Simple, fast, efficient
- Not applicable if no such information available

Explaining Black-box Model

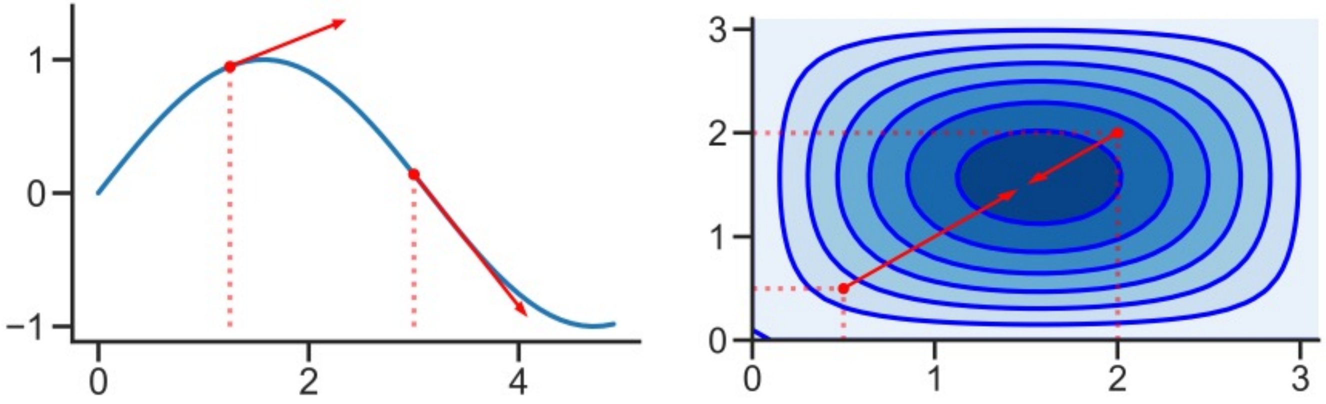
- Gradient-based methods

- Attention-based methods

Gradient-based Explanation

The gradient of a function f on $\boldsymbol{x} \in \mathbb{R}^n$ is

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}$$



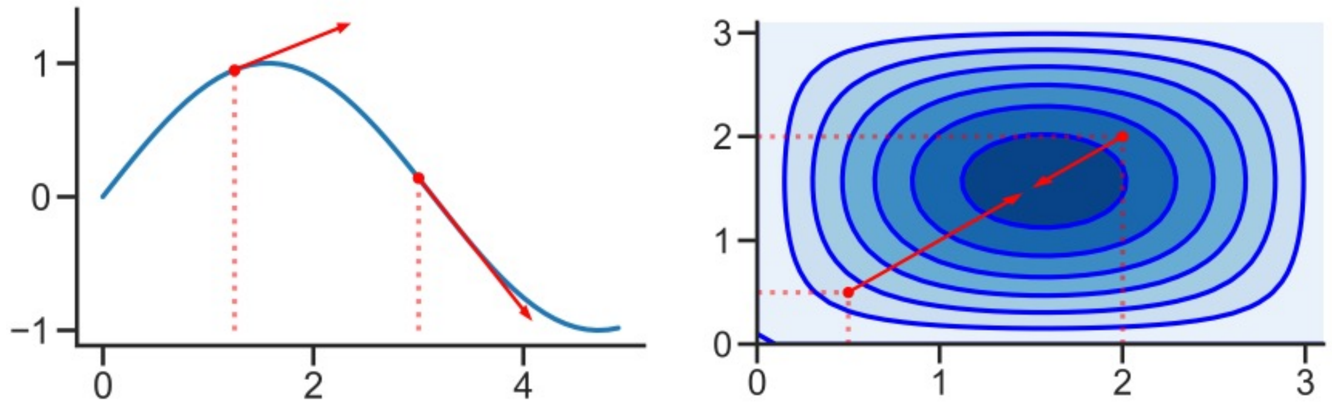
Source: https://towardsdatascience.com/basics-gradient-input-as-explanation-bca79bb80de0

Gradient-based Explanation

The gradient of a function f on $\boldsymbol{x} \in \mathbb{R}^n$ is

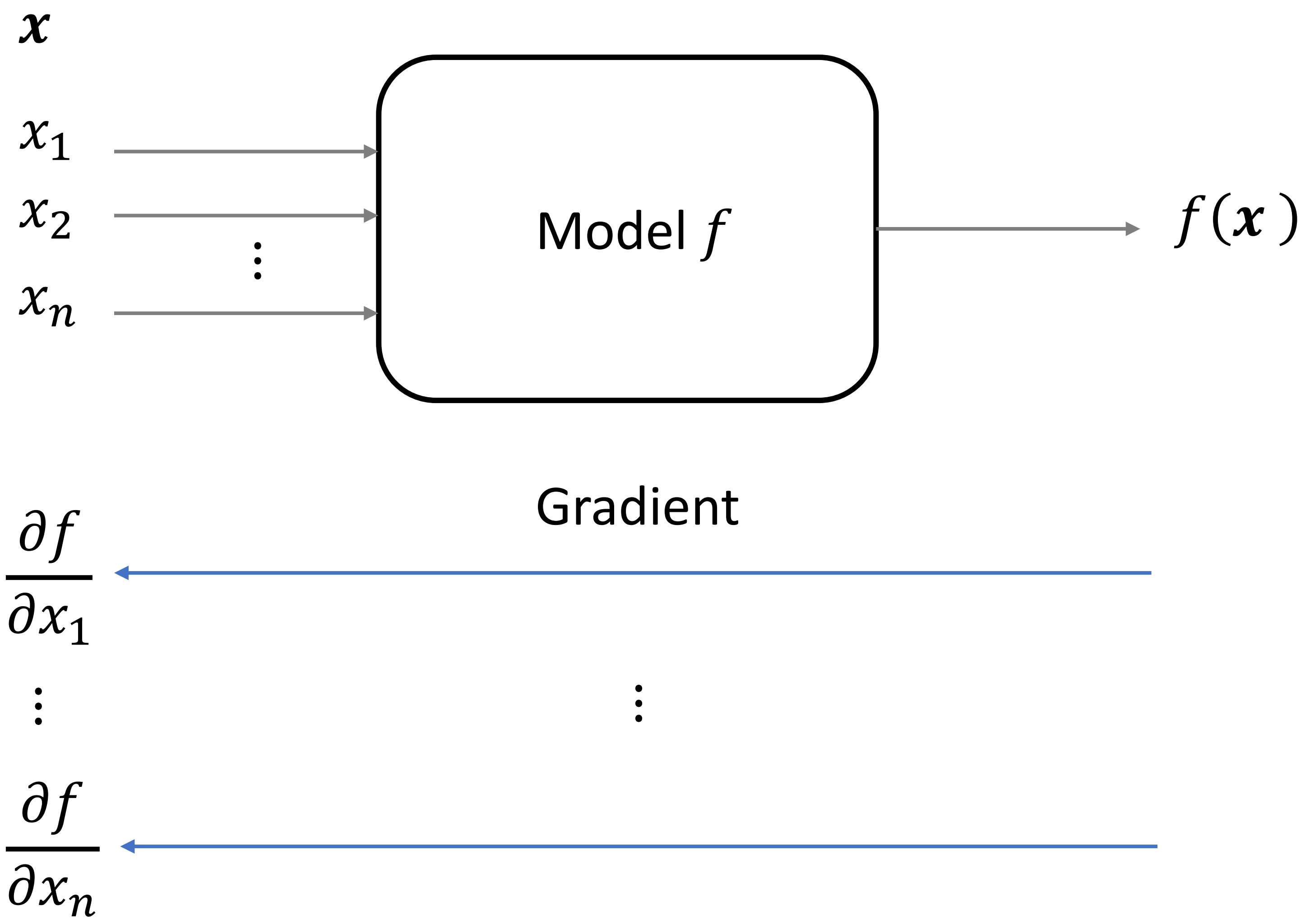
$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}$$

The derivative $\frac{\partial f}{\partial x_i}$ indicates how much f will change when x_i increases a little bit

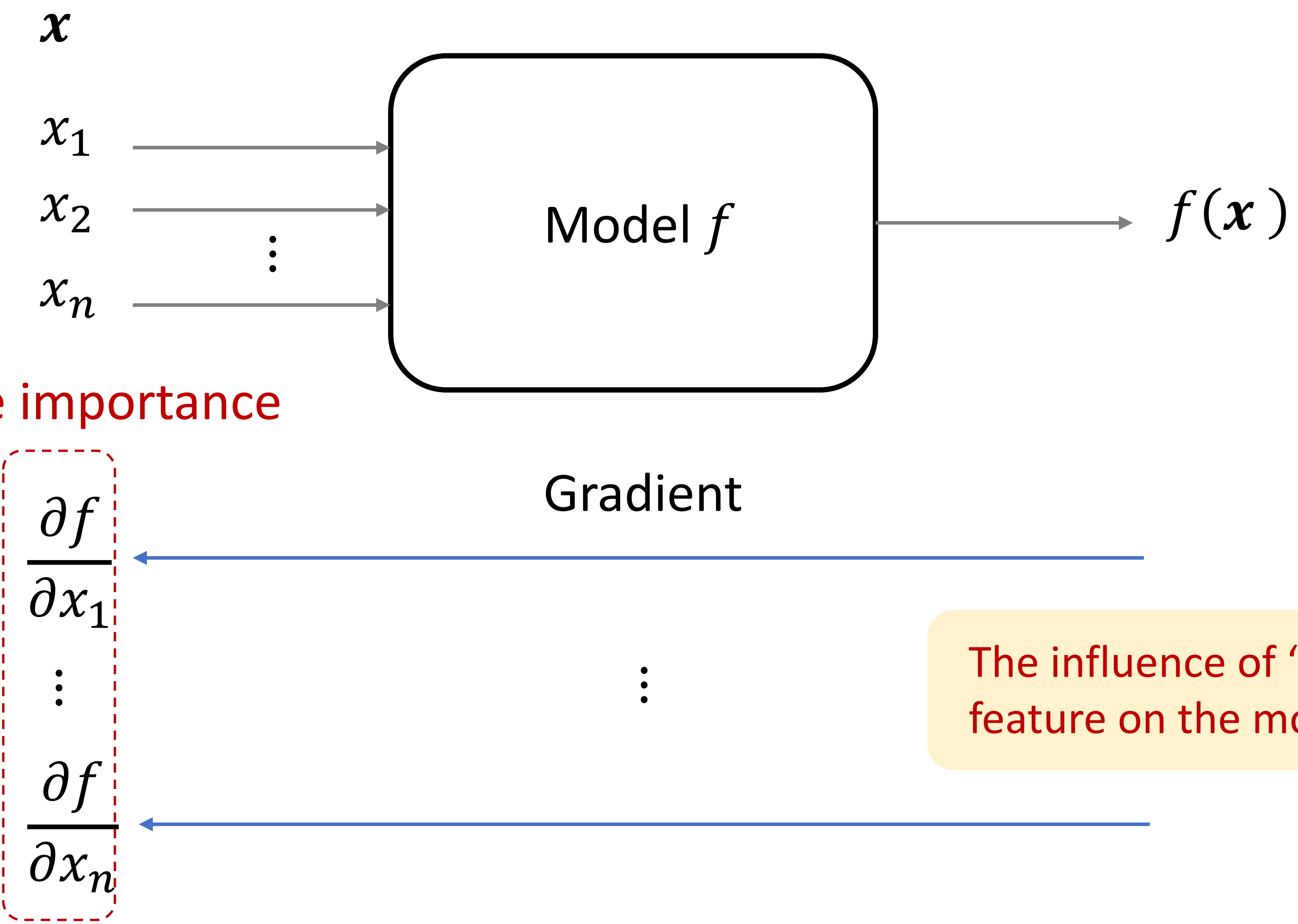


Source: https://towardsdatascience.com/basics-gradient-input-as-explanation-bca79bb80de0

Gradient-based Explanation



Gradient-based Explanation



$\boldsymbol{x}$

x_1

x_2

$\vdots$

x_n

Model f

$f(\boldsymbol{x})$

Feature importance

Gradient

$$\frac{\partial f}{\partial x_1}$$

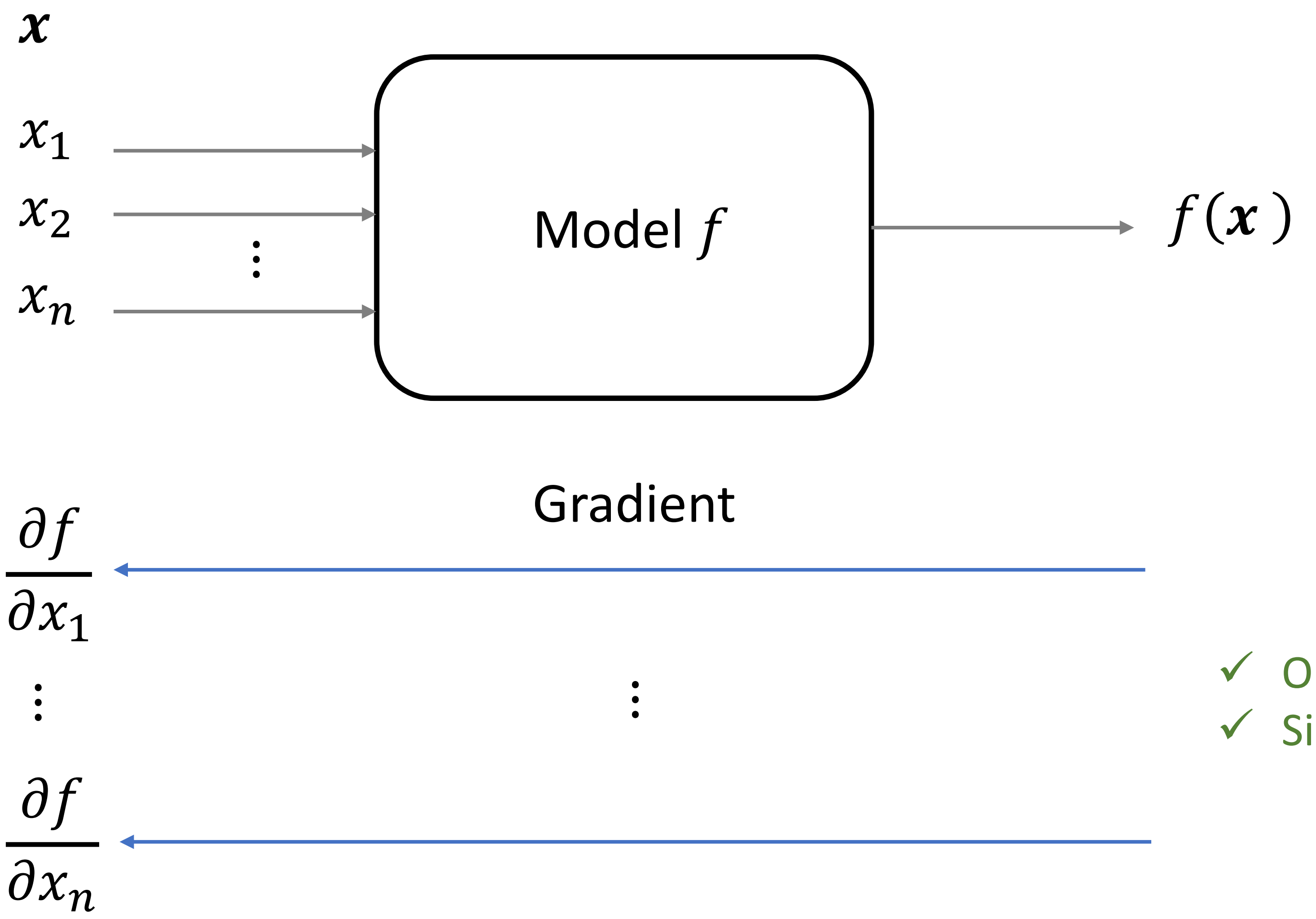
$\vdots$

$\vdots$

$$\frac{\partial f}{\partial x_n}$$

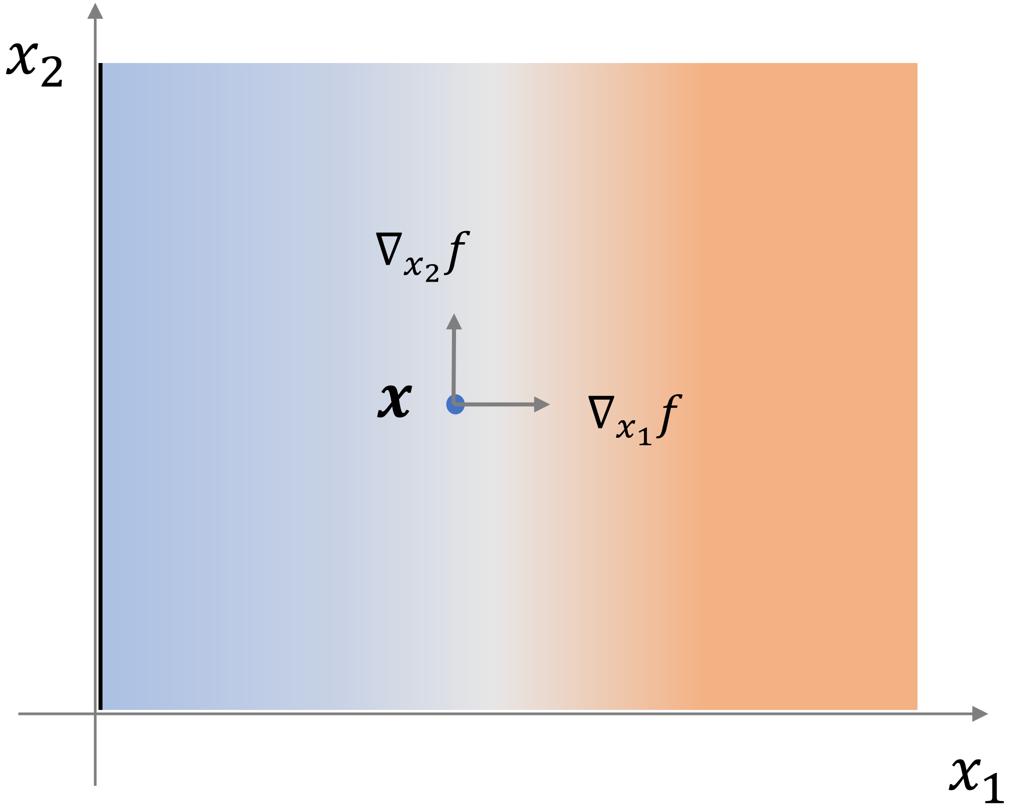
The influence of “tiny change” to the feature on the model prediction

Gradient-based Explanation



- ✓ One backpropagation
- ✓ Simple, fast

Gradient-based Explanation



x_1 is more important than x_2

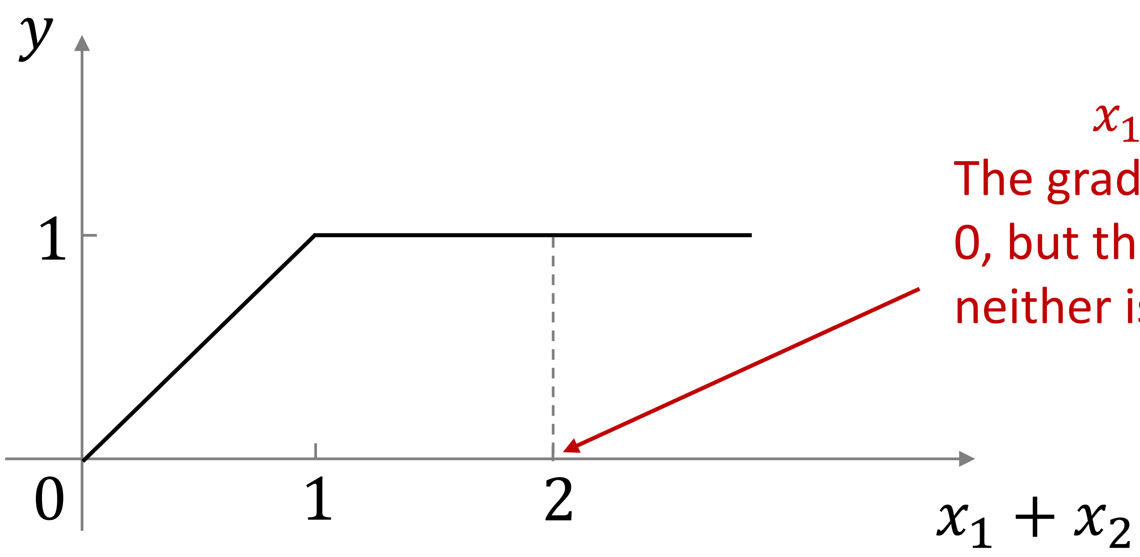
✓ Changing x_1 can flip the model prediction

✓ Changing x_2 would not influence the model prediction

Gradient-based Explanation

Problem 1: saturated outputs lead to unintuitive gradients

$$y = \begin{cases} x_1 + x_2, & \mathit{when}\ (x_1 + x_2) < 1 \\ 1, & \mathit{when}\ (x_1 + x_2) \geq 1 \end{cases}$$



$x_1 = 1, x_2 = 1$

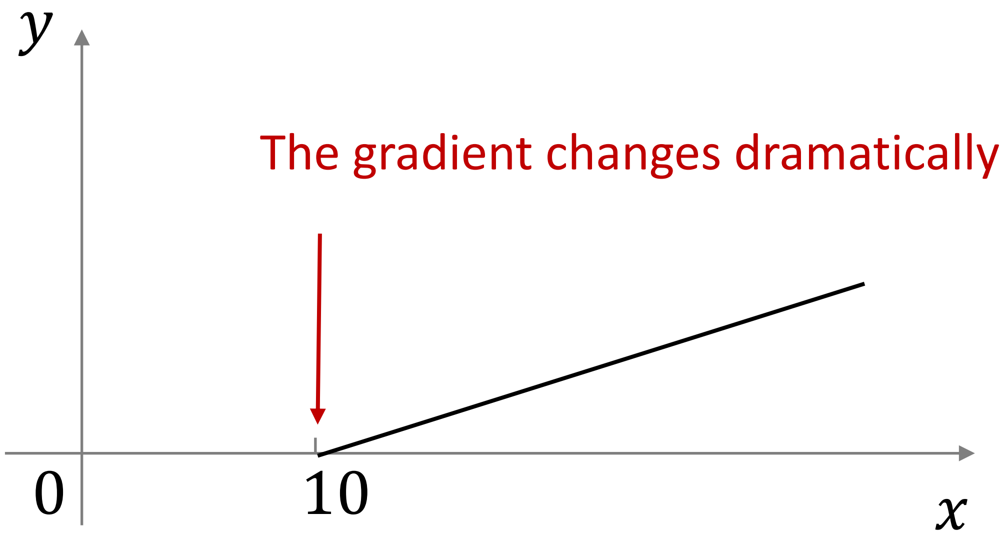
The gradient on x_1 or x_2 is 0, but that does not mean neither is important

(Shrikumar et al., 2017)

Gradient-based Explanation

Problem 2: discontinuous gradients (e.g., thresholding) are problematic

$$y = max(0, x - 10)$$

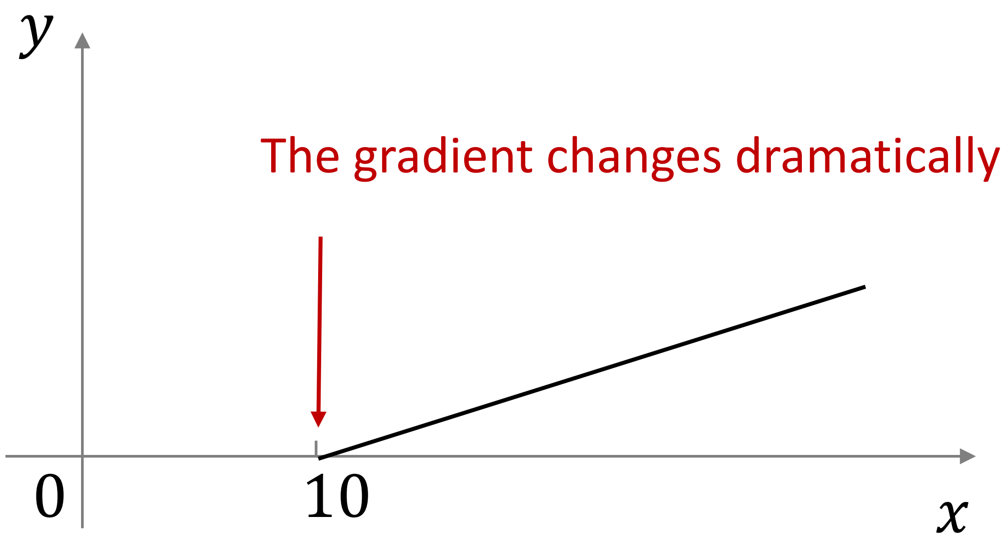


(Shrikumar et al., 2017)

Gradient-based Explanation

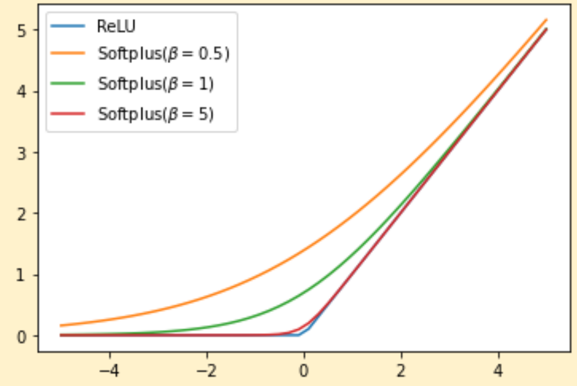
Problem 2: discontinuous gradients (e.g., thresholding) are problematic

$$y = max(0, x - 10)$$



The gradient changes dramatically

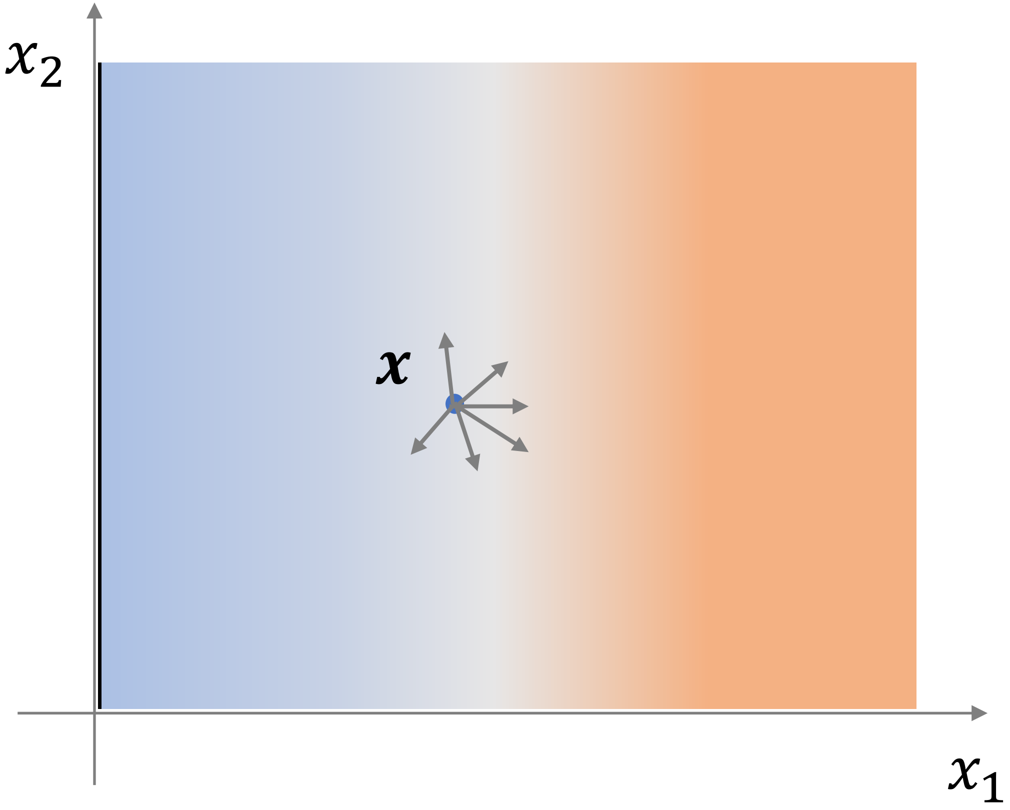
Need to replace “Relu” with “Softplus” activation



(Shrikumar et al., 2017)

Gradient-based Explanation

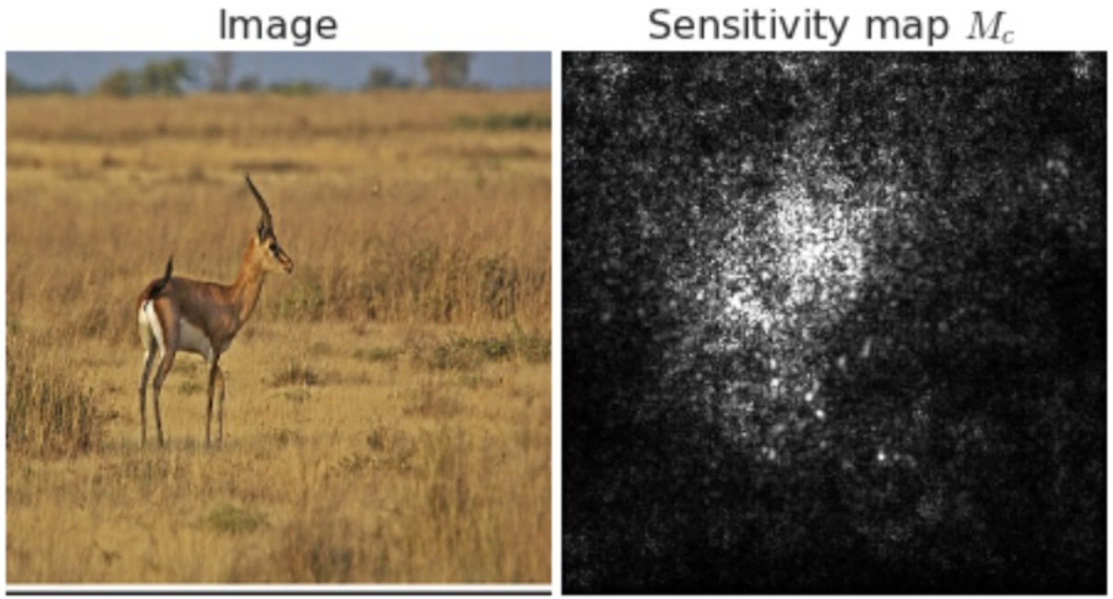
Problem 3: input gradient is sensitive to slight perturbations



Gradient-based Explanation

Problem 3: input gradient is sensitive to slight perturbations

Input gradients are misleading, resulting in a noisy saliency map



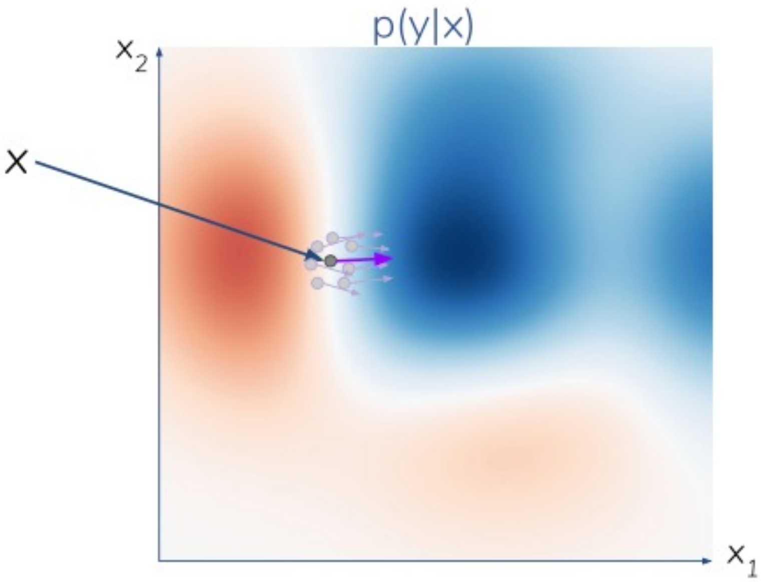
(Smilkov et al., 2017)

Gradient-based Explanation

Do NOT rely on a single gradient calculation

- SmoothGrad: add gaussian noise to inputs and average the gradients

(Smilkov et al., 2017)



Source: EMNLP 2020 Tutorial on Interpreting Predictions of NLP Models

Gradient-based Explanation

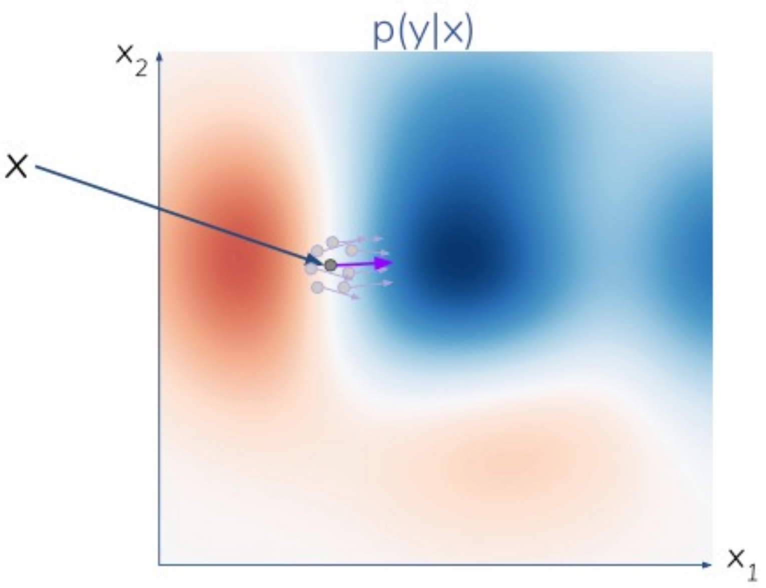
Do NOT rely on a single gradient calculation

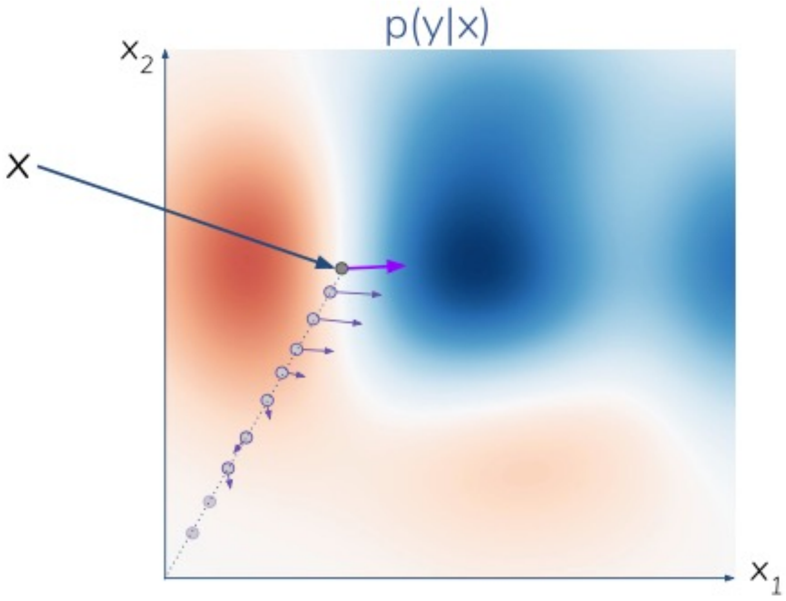
- SmoothGrad: add gaussian noise to inputs and average the gradients

(Smilkov et al., 2017)

- Integrated Gradients: average gradients along a path from baseline to the input

(Sundararajan et al., 2017)





Source: EMNLP 2020 Tutorial on Interpreting Predictions of NLP Models

Gradient-based Explanation

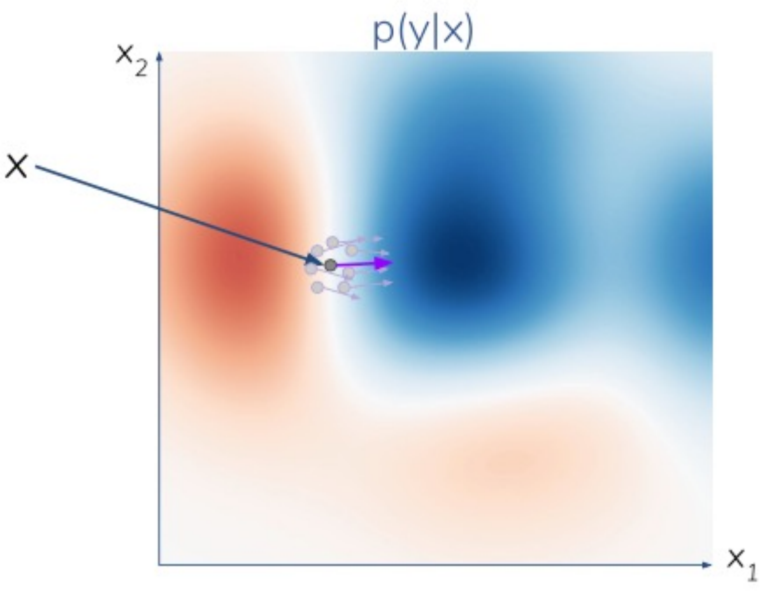
Do NOT rely on a single gradient calculation

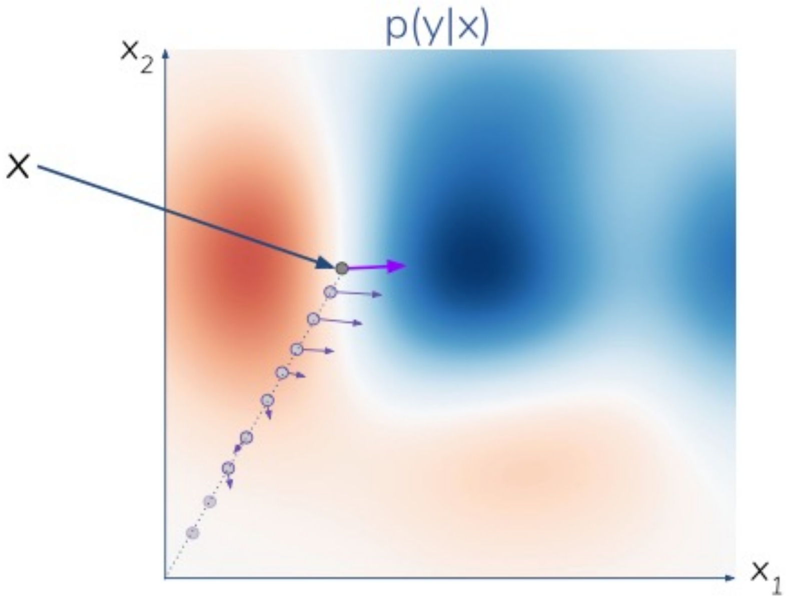
- SmoothGrad: add gaussian noise to inputs and average the gradients

(Smilkov et al., 2017)

- Integrated Gradients: average gradients along a path from baseline to the input

(Sundararajan et al., 2017)





Source: EMNLP 2020 Tutorial on Interpreting Predictions of NLP Models

IG

Axiomatic Attribution for Deep Networks

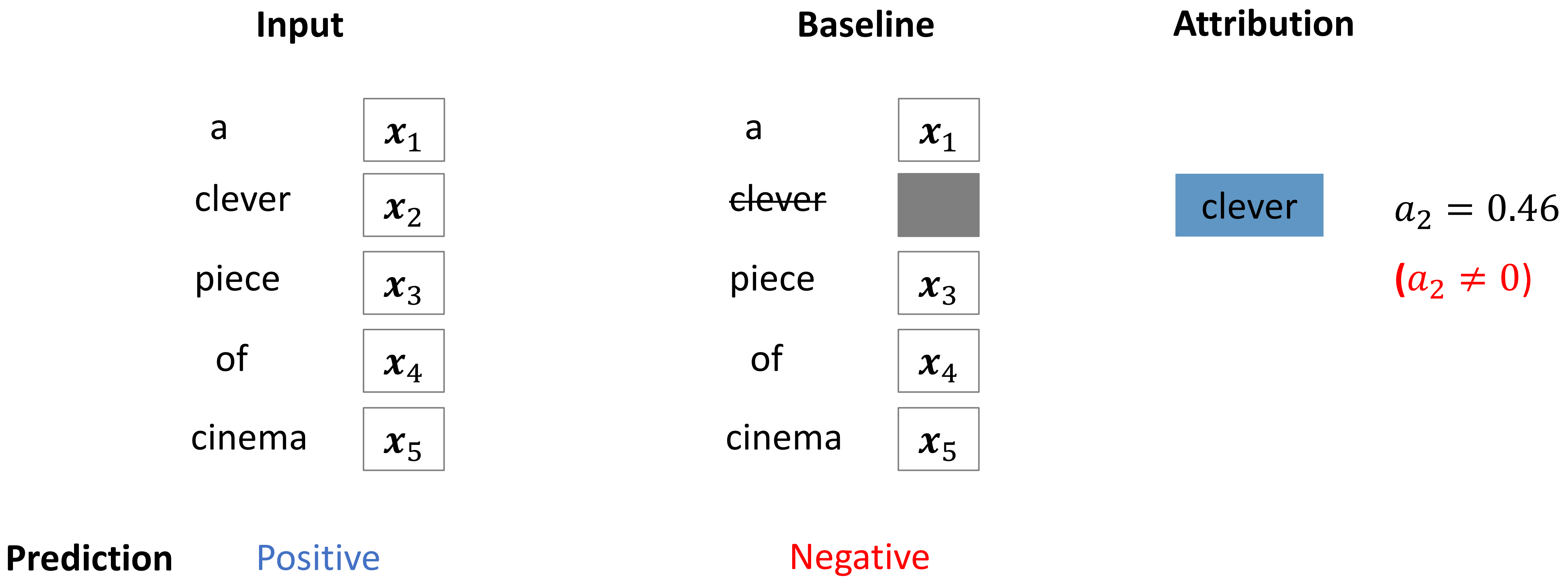
Mukund Sundararajan, Ankur Taly, Qiqi Yan

(ICML, 2017)

Two Fundamental Axioms

- Sensitivity

  For every input and baseline that differ in one feature but have different predictions then the differing feature should be given a non-zero attribution



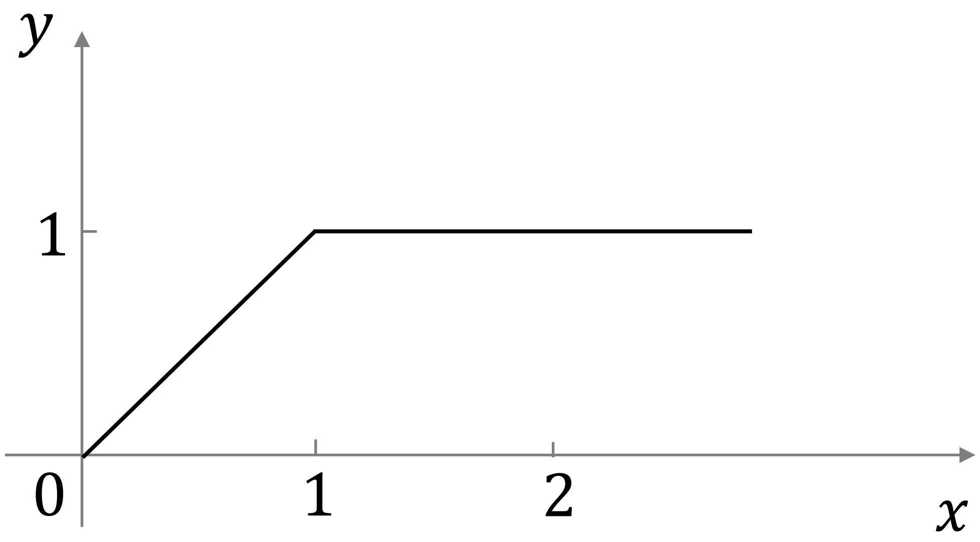
	Input		Baseline		Attribution	
	a	x_1	a	x_1		
	clever	x_2	~~clever~~	▨	clever	$a_2 = 0.46$
	piece	x_3	piece	x_3		$(a_2 \neq 0)$
	of	x_4	of	x_4		
	cinema	x_5	cinema	x_5		
Prediction	Positive		Negative			

Two Fundamental Axioms

- Sensitivity

Gradients violate Sensitivity

$$y = \begin{cases} x, & when\ x < 1 \\ 1, & when\ x \geq 1 \end{cases}$$



Input	Output
$x = 2$	$y = 1$
Baseline	
$x = 0$	$y = 0$

The output changes 1, while the gradient method gives attribution of 0 to x

Two Fundamental Axioms

- Implementation invariance

The attributions are always identical for two functionally equivalent networks

The outputs of two networks are equal for all inputs, despite having very different implementations

$$f\left(h_1(x)\right) = f\left(h_2(x)\right)$$

Two Fundamental Axioms

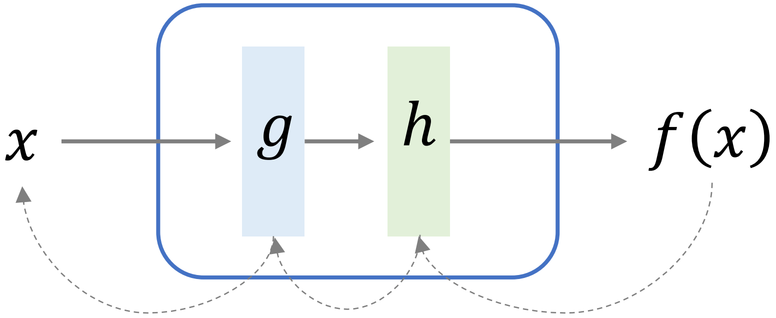
- Implementation invariance

The attributions are always identical for two functionally equivalent networks

✓ Gradients are invariant to implementation

The chain-rule for gradients is essentially about implementation invariance:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial h} \cdot \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial x}$$



Two Fundamental Axioms

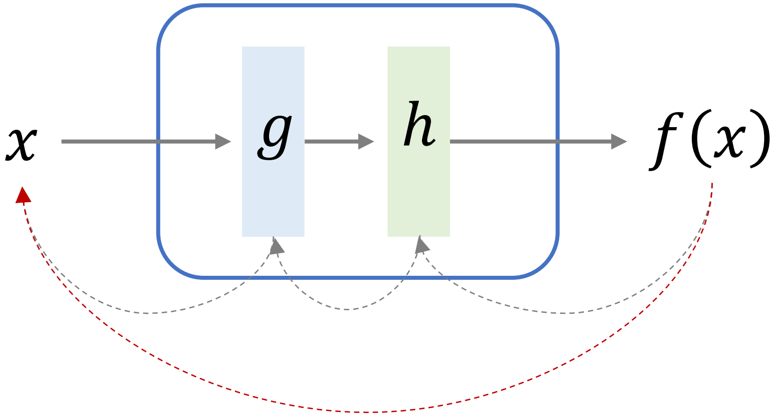
- Implementation invariance

The attributions are always identical for two functionally equivalent networks

✓ Gradients are invariant to implementation

The chain-rule for gradients is essential for implementation invariance:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\cancel{\partial h}} \cdot \frac{\cancel{\partial h}}{\cancel{\partial g}} \cdot \frac{\cancel{\partial g}}{\partial x}$$



Two Fundamental Axioms

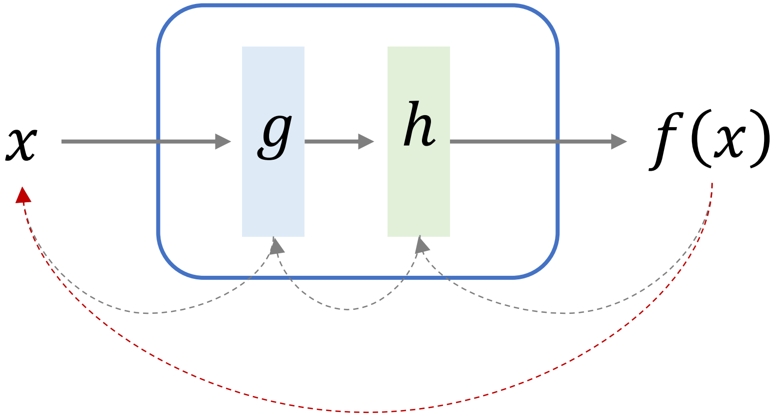
- Implementation invariance

The attributions are always identical for two functionally equivalent networks

✓ Gradients are invariant to implementation

The chain-rule for gradients is essential for implementation invariance:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\cancel{\partial h}} \cdot \frac{\cancel{\partial h}}{\cancel{\partial g}} \cdot \frac{\cancel{\partial g}}{\partial x}$$



✗ Some methods (e.g., LRP and DeepLift) do not satisfy the implementation invariance

IG

- Integrated Gradients

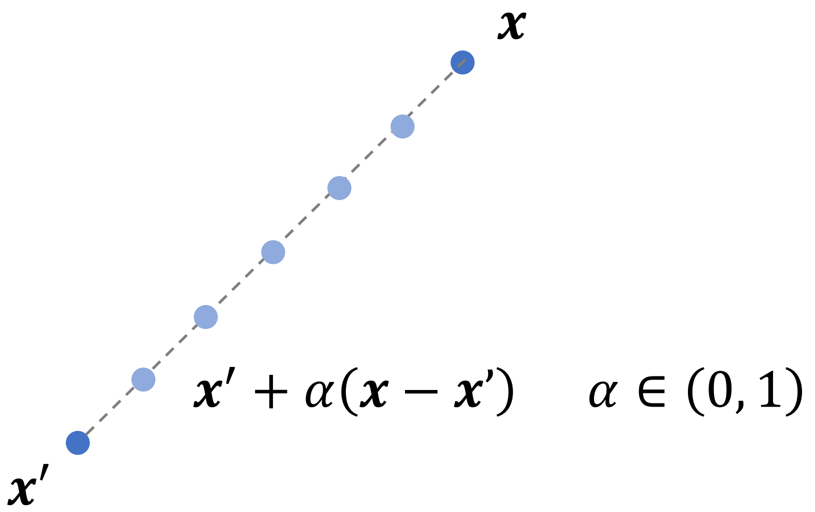
f: neural network

$\boldsymbol{x} \in \mathbb{R}^n$: input

$\boldsymbol{x}' \in \mathbb{R}^n$: baseline

(e.g., black image, zero embedding vector)

Get samples along the straight line from x' to x


$\boldsymbol{x}$
$\boldsymbol{x}' + \alpha(\boldsymbol{x} - \boldsymbol{x}')$ $\alpha \in (0, 1)$
$\boldsymbol{x}'$

IG

- Integrated Gradients

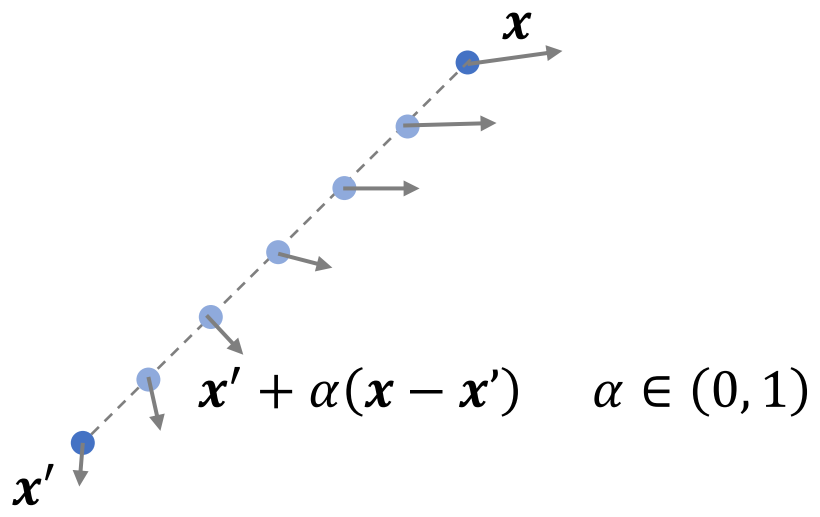
f: neural network

$\boldsymbol{x} \in \mathbb{R}^n$: input

$\boldsymbol{x}' \in \mathbb{R}^n$: baseline

(e.g., black image, zero embedding vector)

Compute gradients at all points along the path



$\boldsymbol{x}' + \alpha(\boldsymbol{x} - \boldsymbol{x}') \quad \alpha \in (0, 1)$

IG

- Integrated Gradients

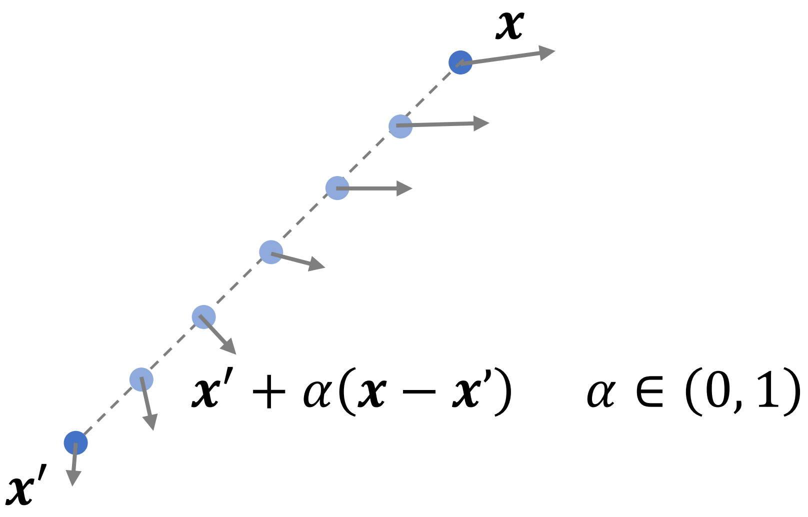
f: neural network

$\boldsymbol{x} \in \mathbb{R}^n$: input

$\boldsymbol{x}' \in \mathbb{R}^n$: baseline

(e.g., black image, zero embedding vector)

Cumulate these gradients


$\boldsymbol{x}$
$\boldsymbol{x}' + \alpha(\boldsymbol{x} - \boldsymbol{x}')$
$\alpha \in (0, 1)$
$\boldsymbol{x}'$

$$IG_{\underline{i}}(\boldsymbol{x}) = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial f\big(\boldsymbol{x}' + \alpha(\boldsymbol{x} - \boldsymbol{x}')\big)}{\partial x_i} d\alpha$$

On the i^{th} dimension

IG

- Integrated Gradients

Axiom: completeness

The attributions add up to the difference between the output of f at the input $\boldsymbol{x}$ and the baseline $\boldsymbol{x}'$

$$\sum_{i=1}^{n} IG_i(\boldsymbol{x}) = f(\boldsymbol{x}) - f(\boldsymbol{x}')$$

IG

- Integrated Gradients

Axiom: completeness

The attributions add up to the difference between the output of f at the input $\boldsymbol{x}$ and the baseline $\boldsymbol{x}'$

$$\sum_{i=1}^{n} IG_i(\boldsymbol{x}) = f(\boldsymbol{x}) - f(\boldsymbol{x}')$$

Sensitivity: for every input and baseline that differ in one feature but have different predictions then the differing feature should be given a non-zero attribution

IG

- Integrated Gradients

Axiom: completeness

The attributions add up to the difference between the output of f at the input $\boldsymbol{x}$ and the baseline $\boldsymbol{x}'$

$$\sum_{i=1}^{n} IG_i(\boldsymbol{x}) = f(\boldsymbol{x}) - f(\boldsymbol{x}')$$

Sensitivity: for every input and baseline that differ in one feature but have different predictions then the differing feature should be given a non-zero attribution

✓ Sensitivity

✓ Implementation invariance

IG

- Integrated Gradients

Axiom: completeness

The attributions add up to the difference between the output of f at the input $\boldsymbol{x}$ and the baseline $\boldsymbol{x}'$

$$\sum_{i=1}^{n} IG_i(\boldsymbol{x}) = f(\boldsymbol{x}) - \underset{f(\boldsymbol{x}') \approx 0}{\underline{f(\boldsymbol{x}')}}$$

Shapley

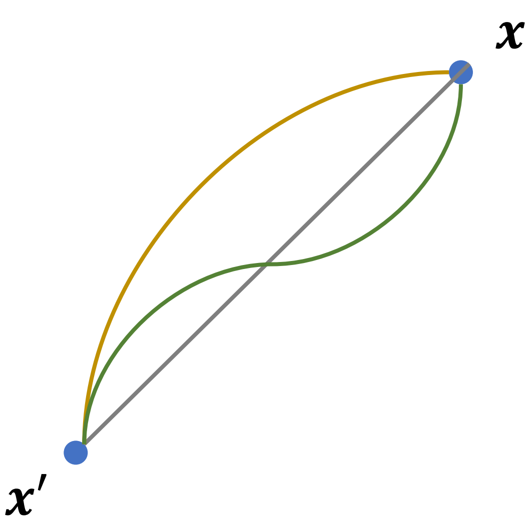
$$g(z) = \phi_0 + \sum_{i=1}^{n} \phi_i z_i$$

Question?

IG

- Uniqueness of Integrated Gradients

Each path yields a different attribution method



$$PathIG_i(\boldsymbol{x}) = \int_{\alpha=0}^{1} \frac{\partial f\big(\gamma(\alpha)\big)}{\partial \gamma_i(\alpha)} \frac{\partial \gamma_i(\alpha)}{\partial \alpha} d\alpha$$

$\gamma(\alpha)$: path function, $\gamma(0) = \boldsymbol{x}'$, $\gamma(1) = \boldsymbol{x}$

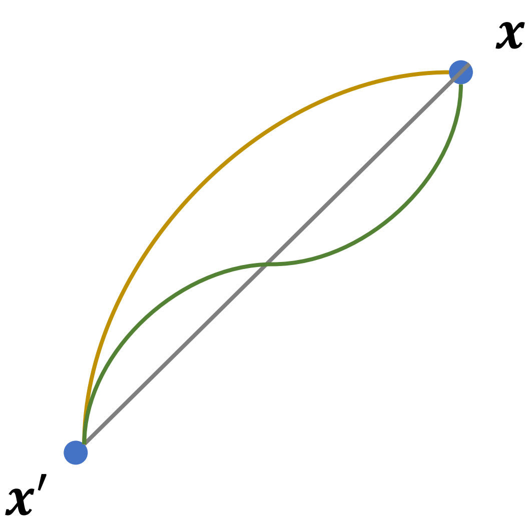
IG is the straight path:

$$\gamma(\alpha) = \boldsymbol{x}' + \alpha(\boldsymbol{x} - \boldsymbol{x}')$$

IG

- Uniqueness of Integrated Gradients

Each path yields a different attribution method



$$PathIG_i(\boldsymbol{x}) = \int_{\alpha=0}^{1} \frac{\partial f\left(\gamma(\alpha)\right)}{\partial \gamma_i(\alpha)} \frac{\partial \gamma_i(\alpha)}{\partial \alpha} d\alpha$$

$\gamma(\alpha)$: path function, $\gamma(0) = \boldsymbol{x}'$, $\gamma(1) = \boldsymbol{x}$

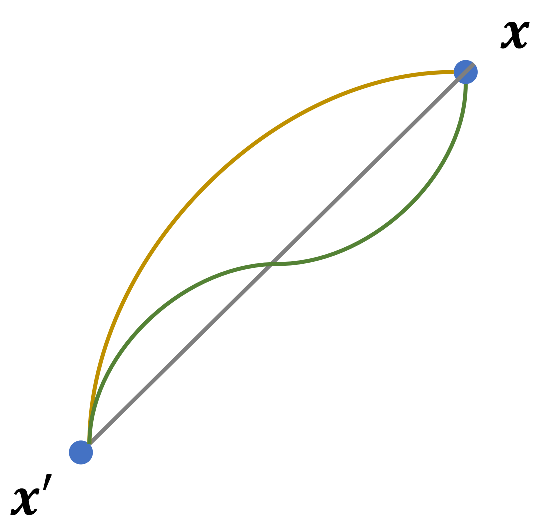
✓ Sensitivity

✓ Implementation invariance

IG

- Uniqueness of Integrated Gradients

Why the straightline path chosen by integrated gradients is canonical?



✓ The simplest path

✓ Preserving symmetry

For all inputs and baselines that have identical values for symmetric variables, the symmetric variables receive identical attributions

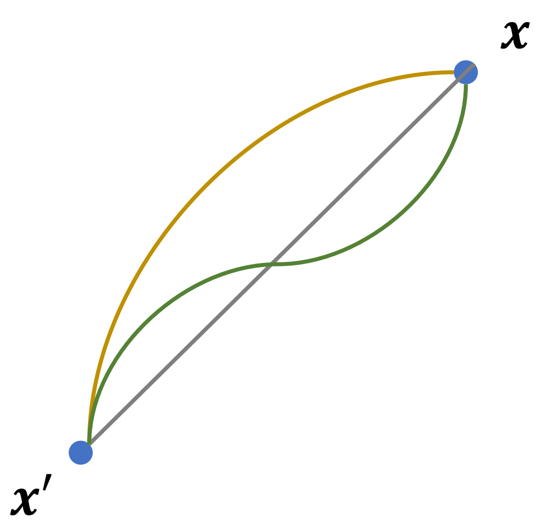
Swapping the two variables does not change the function

$$f(x,y)=f(y,x)$$

IG

- Uniqueness of Integrated Gradients

Why the straightline path chosen by integrated gradients is canonical?



✓ The simplest path
✓ Preserving symmetry

For all inputs and baselines that have identical values for symmetric variables, the symmetric variables receive identical attributions

Example

$$logistic\_regression(x_1 + x_2)$$

Input: $x_1 = x_2 = 1$

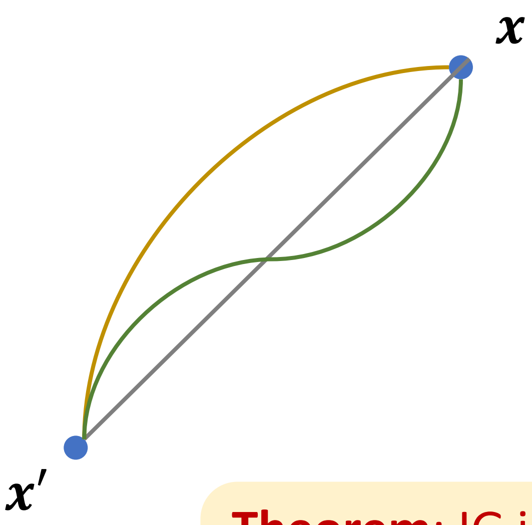
Baseline: $x_1 = x_2 = 0$

$Attr(x_1) = Attr(x_2)$

IG

- Uniqueness of Integrated Gradients

Why the straightline path chosen by integrated gradients is canonical?



Theorem: IG is the unique path method that is symmetry-preserving

✓ The simplest path
✓ Preserving symmetry

For all inputs and baselines that have identical values for symmetric variables, the symmetric variables receive identical attributions

Example

$$logistic\_regression(x_1 + x_2)$$

Input: $x_1 = x_2 = 1$

Baseline: $x_1 = x_2 = 0$

$Attr(x_1) = Attr(x_2)$

IG

- Applying Integrated Gradients

The integral of integrated gradients can be efficiently approximated via a summation

$$IG_i(\boldsymbol{x}) \approx (x_i - x_i{}')\times \sum_{k=1}^{m} \frac{\partial f\left(\boldsymbol{x}' + \frac{k}{m}(\boldsymbol{x} - \boldsymbol{x}')\right)}{\partial x_i} \times \frac{1}{m}$$

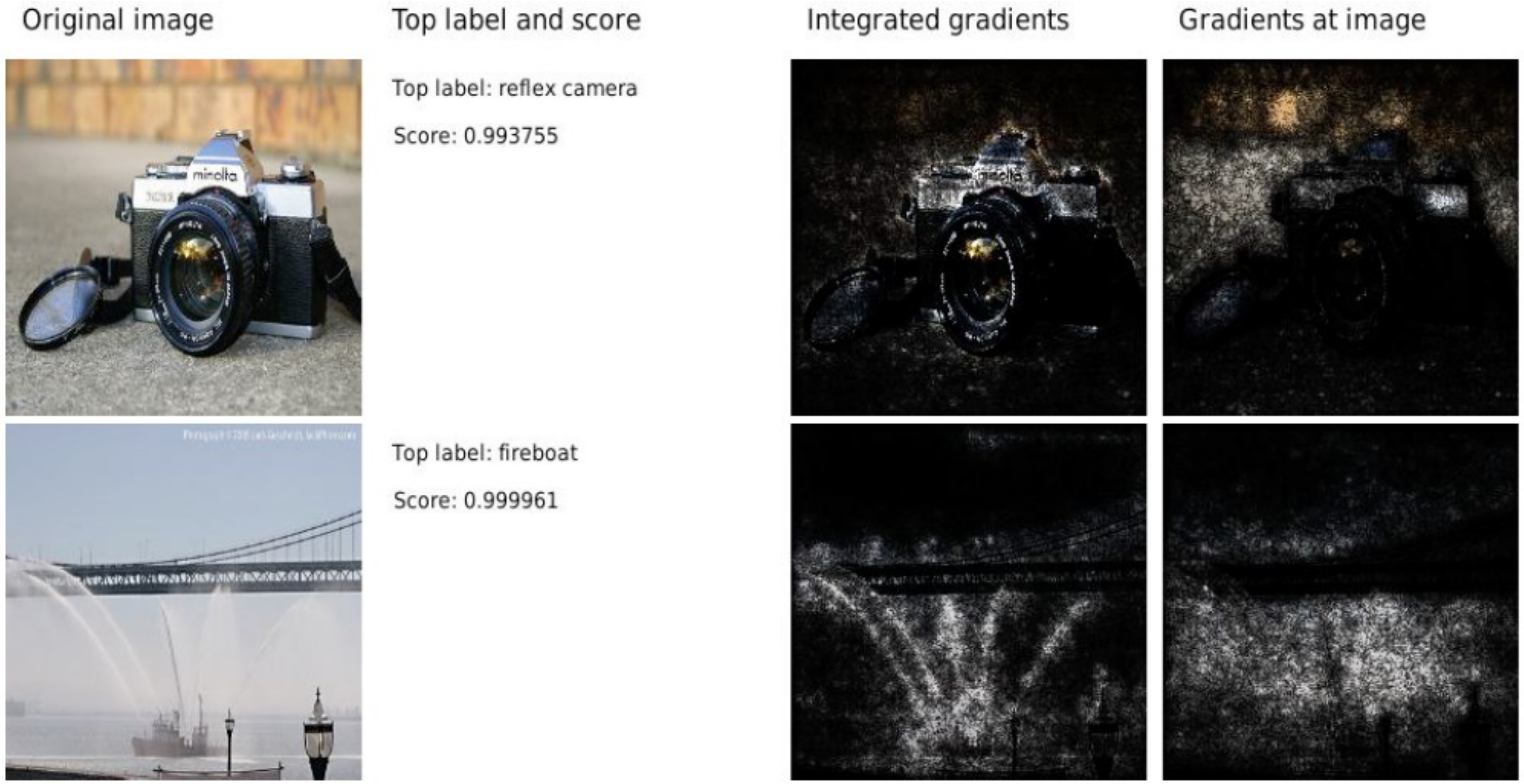
m: the number of steps

IG

- Applications of Integrated Gradients

Task: object recognition
Model: GoogleNet
Dataset: ImageNet

Integrated gradients are better at reflecting distinctive features of the input image



Question?

Explaining Black-box Model

- Gradient-based methods

- Attention-based methods

Attention

What is attention?

In psychology, attention is the cognitive process of selectively concentrating on one or a few things while ignoring others

Source: https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/

Attention

What is attention?

In psychology, attention is the cognitive process of selectively concentrating on one or a few things while ignoring others

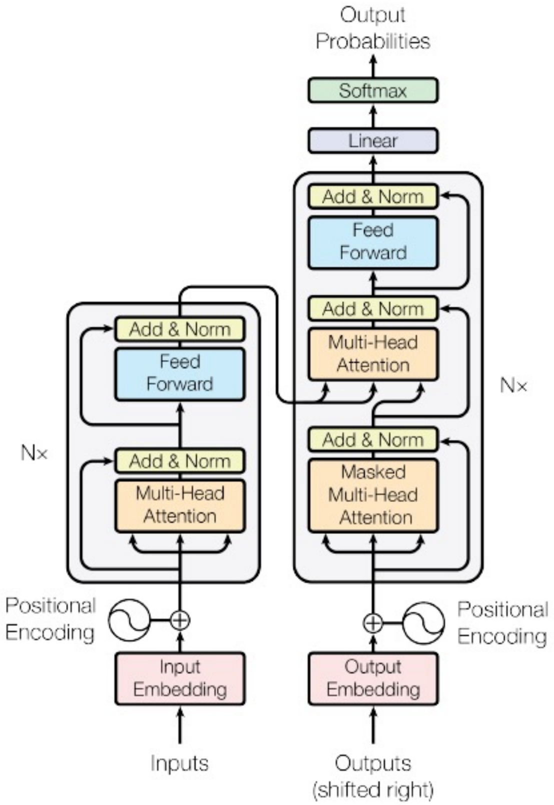
The attention mechanism for neural networks is to mimic human brain actions in a simplified manner

Source: https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/

Attention

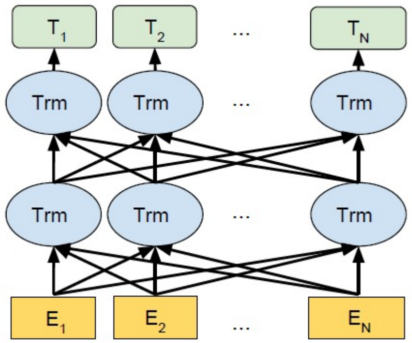
Light up natural language procession (NLP)

Transformer



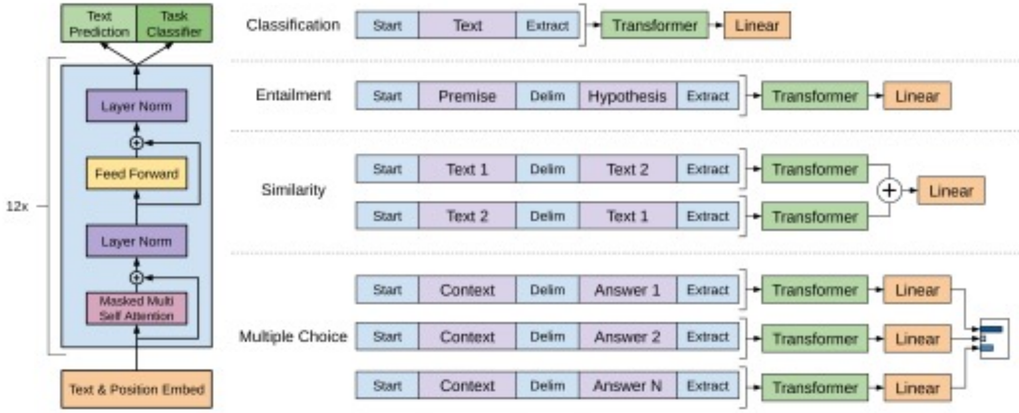
(Vaswani et al., 2017)

BERT



(Devlin et al., 2018)

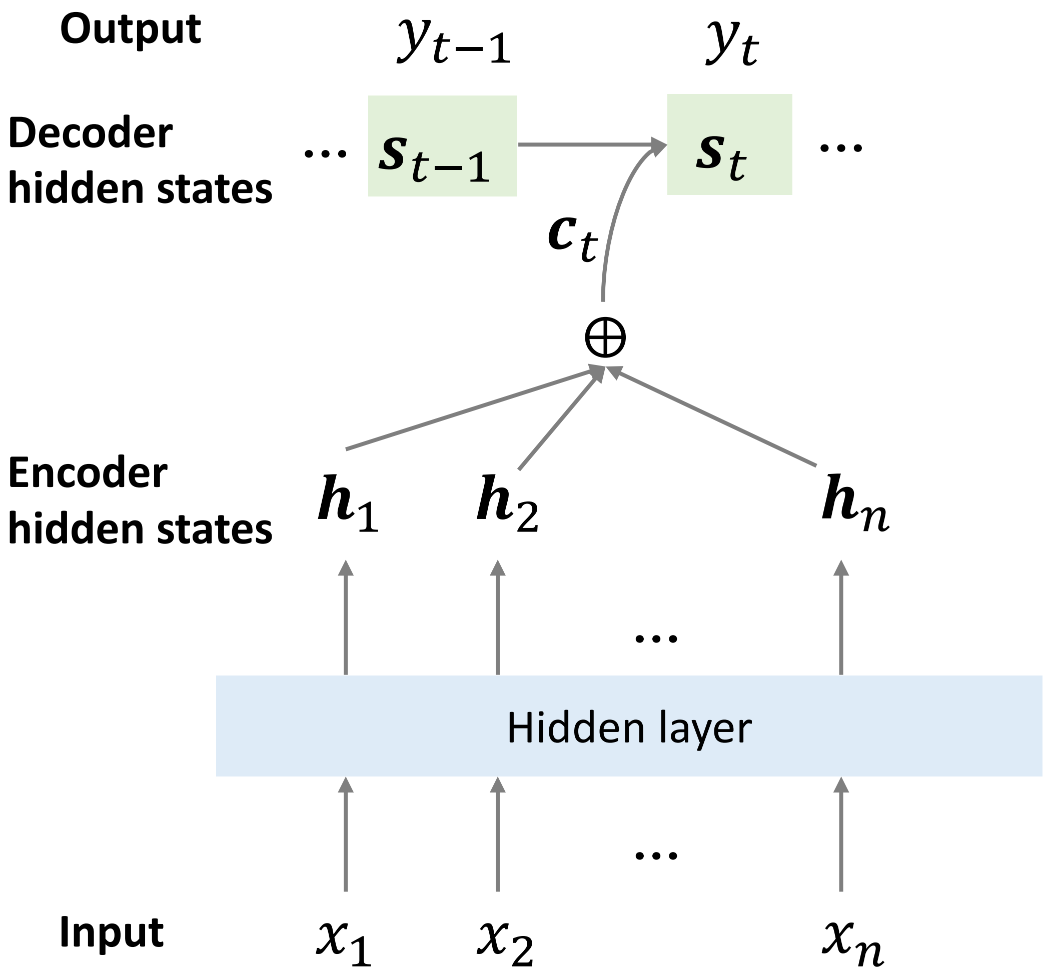
GPT



(Radford et al., 2018)

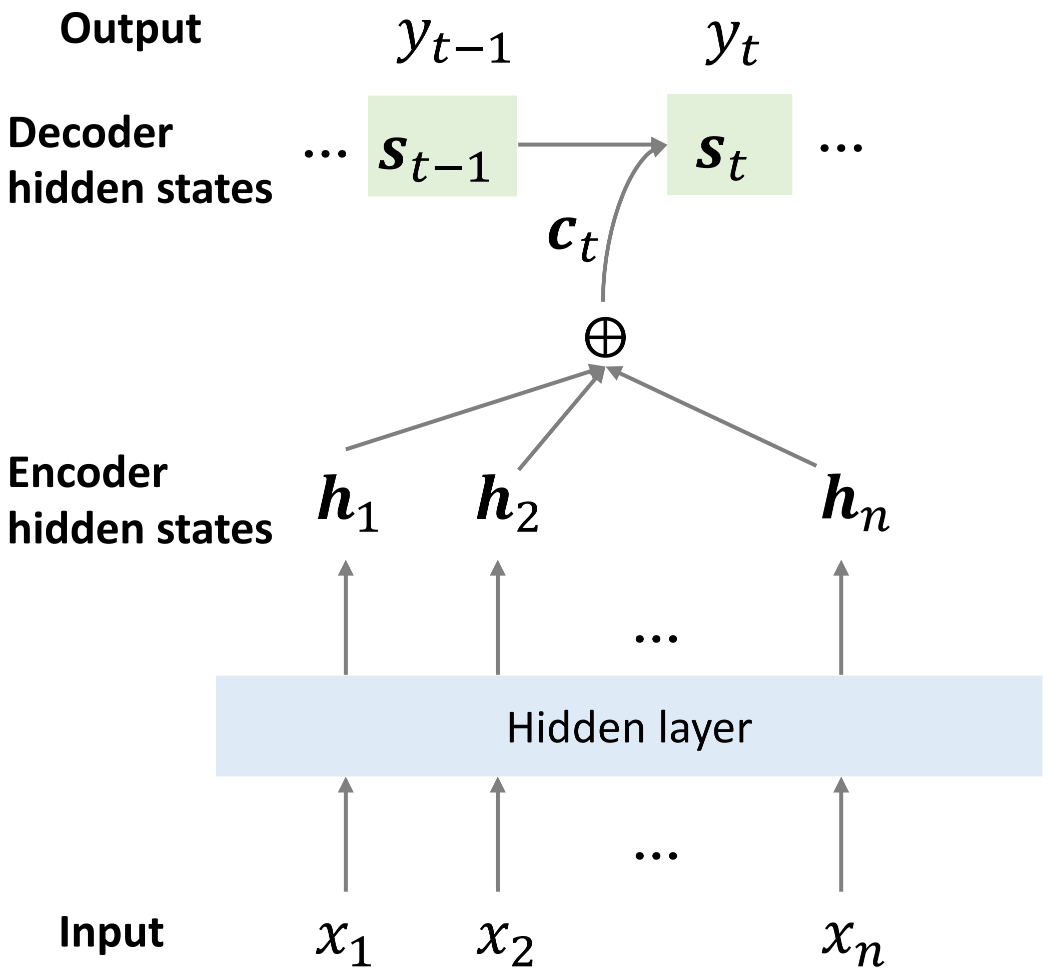
Attention

Context vector: a good summary of the input



Attention

Context vector: a good summary of the input



$$\boldsymbol{c}_t = \sum_{i=1}^{n} \alpha_{ti} \boldsymbol{h}_i$$

Context vector for output y_t

$$\alpha_{ti} = align(y_t, x_i)$$

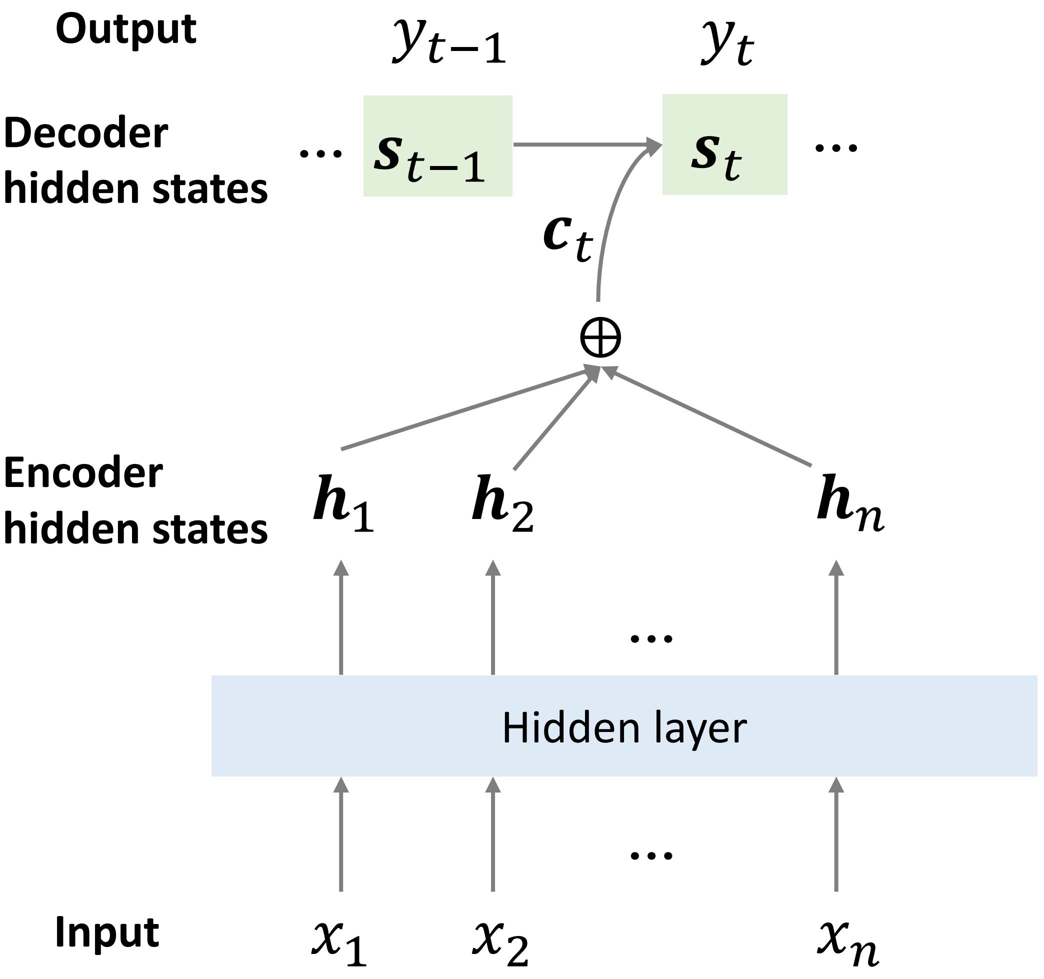
How well y_t and x_i are aligned

$$= \frac{exp\big(score(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i)\big)}{\sum_{k=1}^{n} exp\big(score(\boldsymbol{s}_{t-1}, \boldsymbol{h}_k)\big)}$$

Softmax of some predefined alignment score

Attention

Context vector: a good summary of the input



$$\boldsymbol{c}_t = \sum_{i=1}^{n} \alpha_{ti} \boldsymbol{h}_i$$

Context vector for output y_t

$$\alpha_{ti} = align(y_t, x_i)$$

How well y_t and x_i are aligned

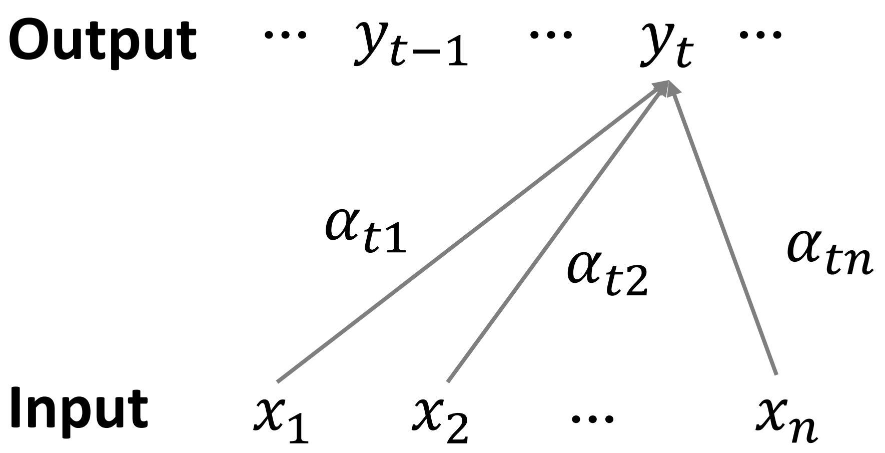
$$= \frac{exp\big(score(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i)\big)}{\sum_{k=1}^{n} exp\big(\underline{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_k)\big)}$$

Softmax of some predefined alignment score

Can be parametrized by a feed-forward network jointly trained with other parts of the model

Attention

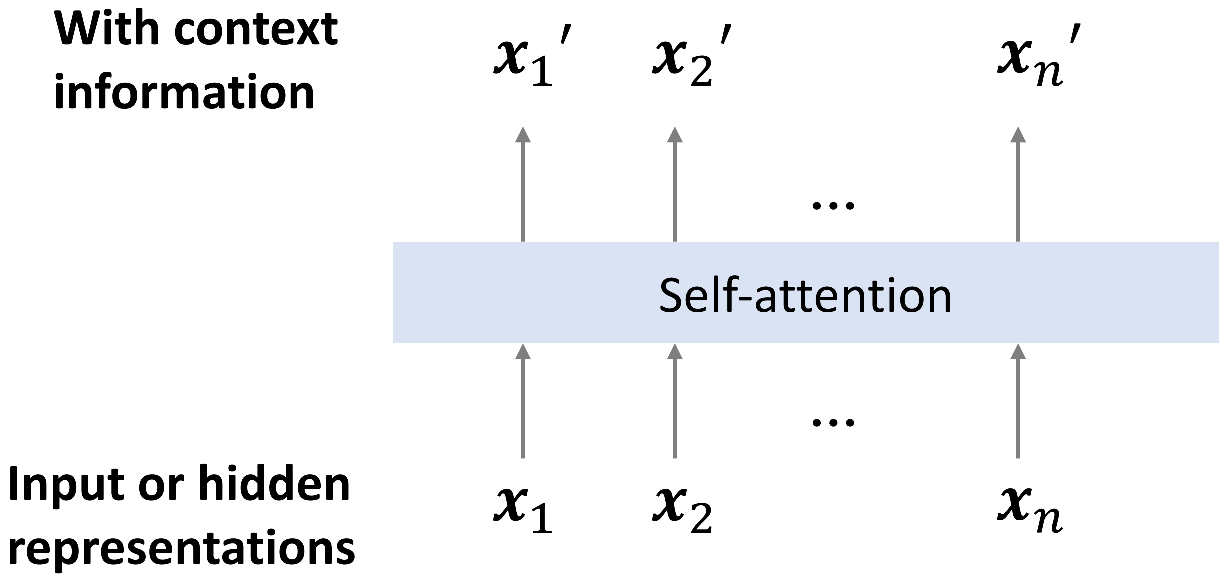
The attention weights $\{\alpha_{ti}\}$ somehow indicate how much of each input feature contributes to each output


Output $\cdots$ y_{t-1} $\cdots$ y_t $\cdots$
α_{t1}
α_{t2}
α_{tn}
Input x_1 x_2 $\cdots$ x_n

- ✓ Simple, fast
- ✓ No additional computation

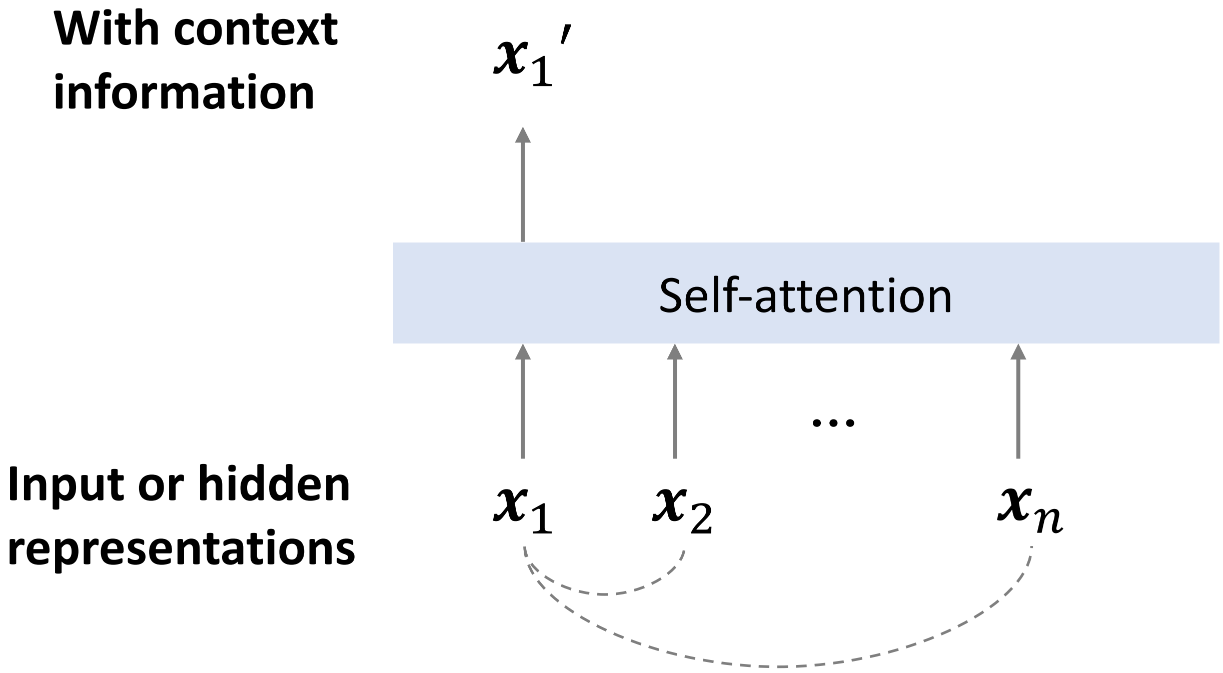
Attention

Self-attention mechanism



Attention

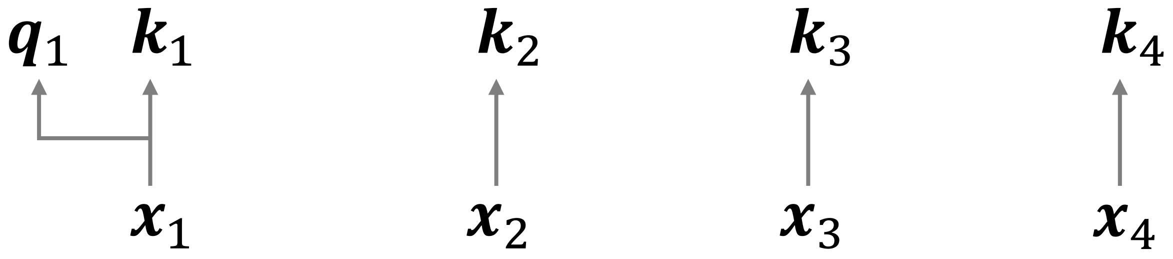
Self-attention mechanism



Attention

Self-attention mechanism

Query: $\boldsymbol{q}$
Key: $\boldsymbol{k}$
Value: $\boldsymbol{v}$



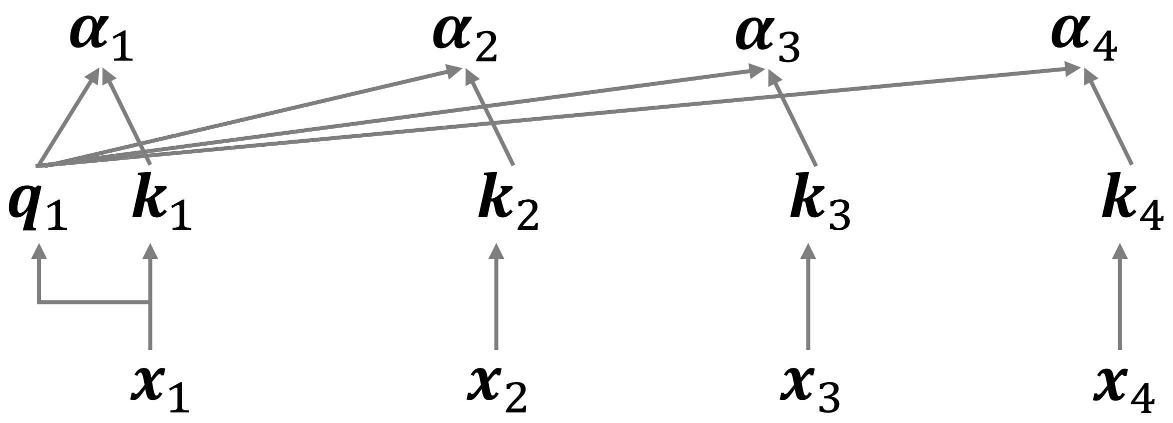
$\boldsymbol{q}_1 = W^q \boldsymbol{x}_1$
$\boldsymbol{k}_1 = W^k \boldsymbol{x}_1$

$\boldsymbol{k}_2 = W^k \boldsymbol{x}_2$ $\boldsymbol{k}_3 = W^k \boldsymbol{x}_3$ $\boldsymbol{k}_4 = W^k \boldsymbol{x}_4$

Attention

Self-attention mechanism

Query: $\boldsymbol{q}$
Key: $\boldsymbol{k}$
Value: $\boldsymbol{v}$

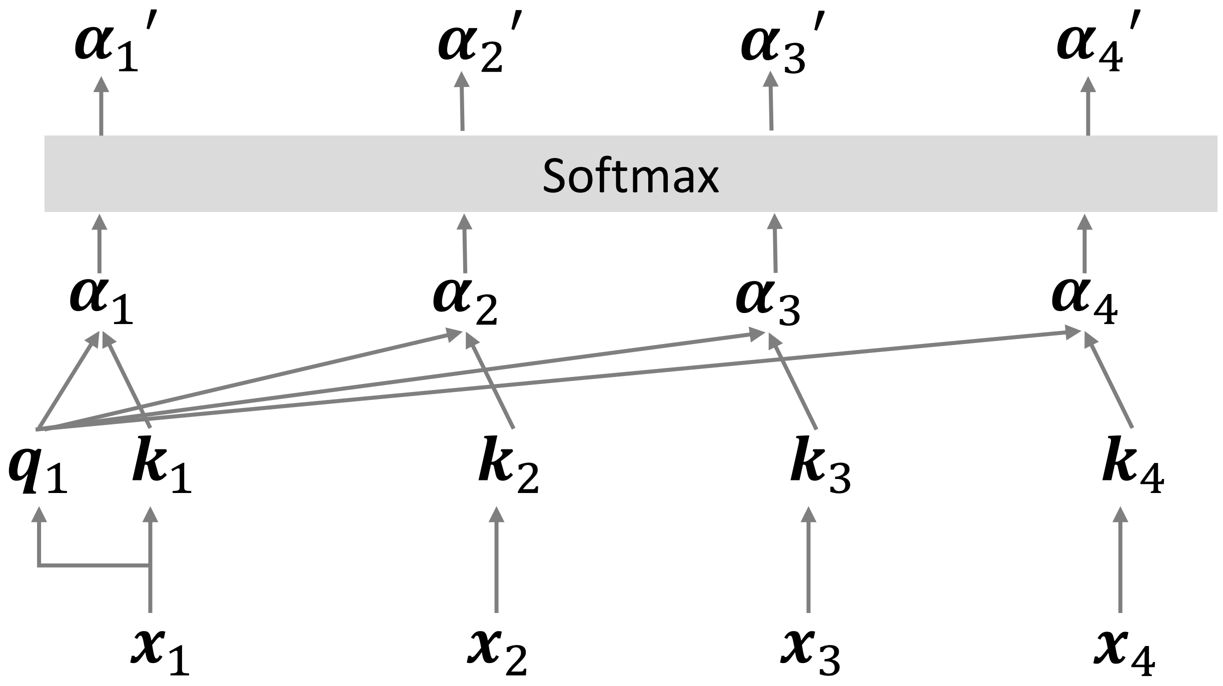


$\boldsymbol{q}_1 = W^q \boldsymbol{x}_1$ $\boldsymbol{k}_2 = W^k \boldsymbol{x}_2$ $\boldsymbol{k}_3 = W^k \boldsymbol{x}_3$ $\boldsymbol{k}_4 = W^k \boldsymbol{x}_4$
$\boldsymbol{k}_1 = W^k \boldsymbol{x}_1$

Attention

Self-attention mechanism

Query: $\boldsymbol{q}$
Key: $\boldsymbol{k}$
Value: $\boldsymbol{v}$



$\boldsymbol{q}_1 = W^q \boldsymbol{x}_1$ $\quad$ $\boldsymbol{k}_2 = W^k \boldsymbol{x}_2$ $\quad$ $\boldsymbol{k}_3 = W^k \boldsymbol{x}_3$ $\quad$ $\boldsymbol{k}_4 = W^k \boldsymbol{x}_4$

$\boldsymbol{k}_1 = W^k \boldsymbol{x}_1$

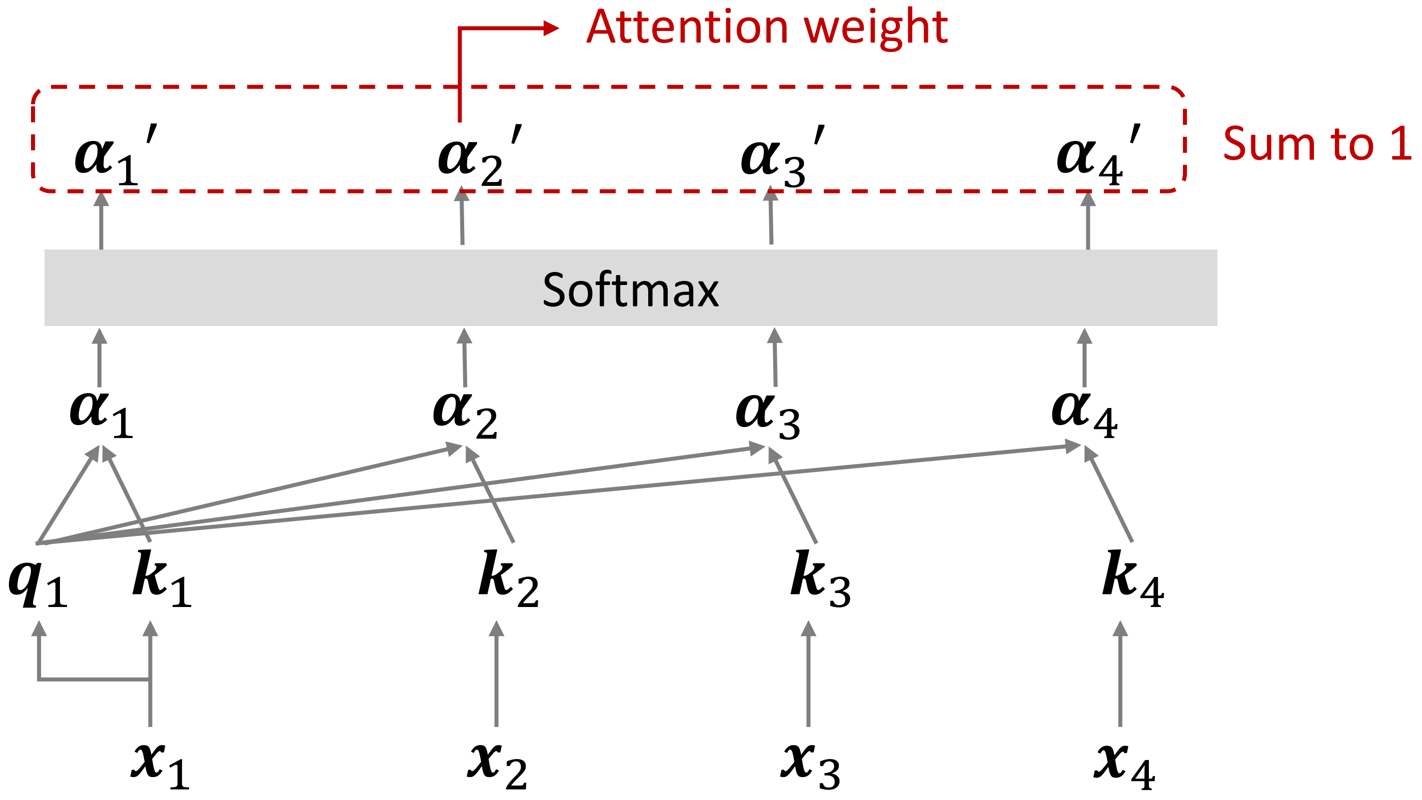
Attention

Self-attention mechanism

Query: $\boldsymbol{q}$

Key: $\boldsymbol{k}$

Value: $\boldsymbol{v}$


Attention weight
Sum to 1
$\boldsymbol{\alpha}_1'$ $\boldsymbol{\alpha}_2'$ $\boldsymbol{\alpha}_3'$ $\boldsymbol{\alpha}_4'$
Softmax
$\boldsymbol{\alpha}_1$ $\boldsymbol{\alpha}_2$ $\boldsymbol{\alpha}_3$ $\boldsymbol{\alpha}_4$
$\boldsymbol{q}_1$ $\boldsymbol{k}_1$ $\boldsymbol{k}_2$ $\boldsymbol{k}_3$ $\boldsymbol{k}_4$
$\boldsymbol{x}_1$ $\boldsymbol{x}_2$ $\boldsymbol{x}_3$ $\boldsymbol{x}_4$

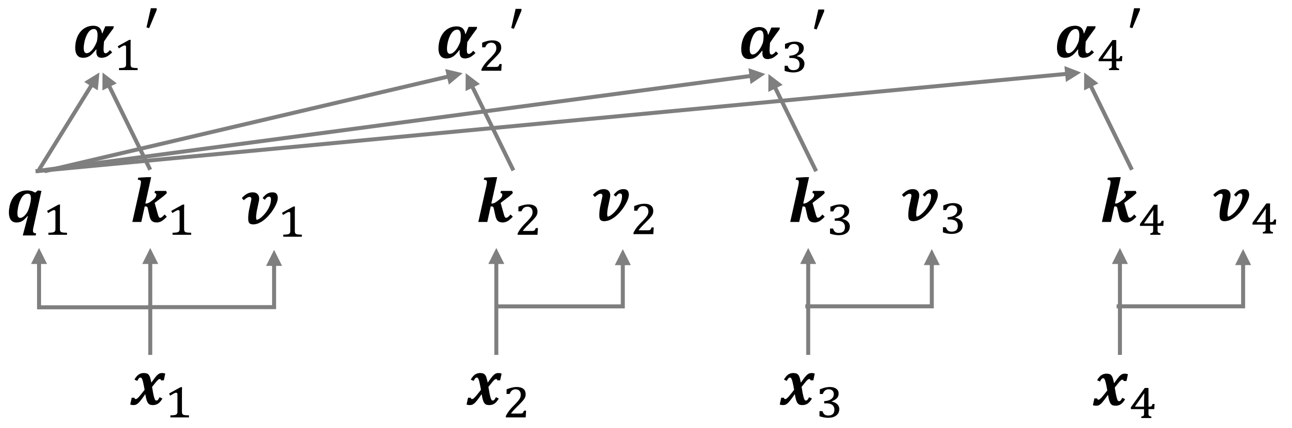
$\boldsymbol{q}_1 = W^q \boldsymbol{x}_1$ $\quad$ $\boldsymbol{k}_2 = W^k \boldsymbol{x}_2$ $\quad$ $\boldsymbol{k}_3 = W^k \boldsymbol{x}_3$ $\quad$ $\boldsymbol{k}_4 = W^k \boldsymbol{x}_4$

$\boldsymbol{k}_1 = W^k \boldsymbol{x}_1$

Attention

Self-attention mechanism

Query: $\boldsymbol{q}$
Key: $\boldsymbol{k}$
Value: $\boldsymbol{v}$



$\boldsymbol{q}_1 = W^q \boldsymbol{x}_1$
$\boldsymbol{k}_1 = W^k \boldsymbol{x}_1$
$\boldsymbol{v}_1 = W^v \boldsymbol{x}_1$

$\boldsymbol{k}_2 = W^k \boldsymbol{x}_2$
$\boldsymbol{v}_2 = W^v \boldsymbol{x}_2$

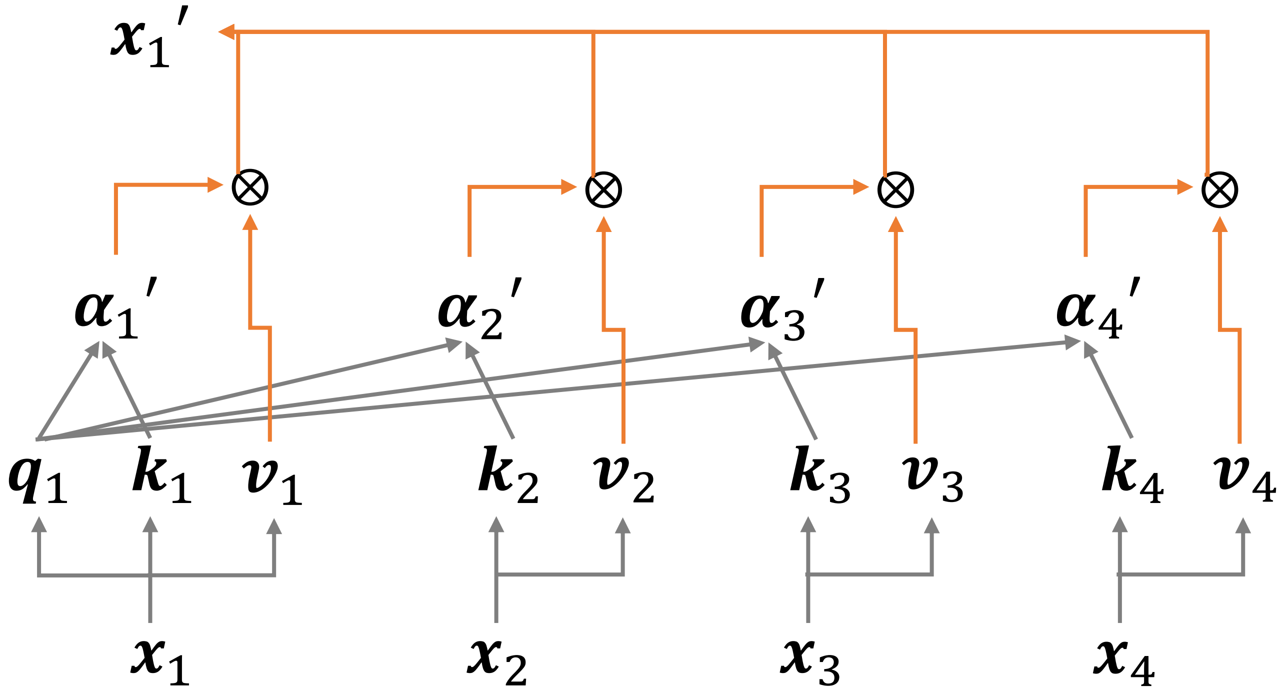
$\boldsymbol{k}_3 = W^k \boldsymbol{x}_3$
$\boldsymbol{v}_3 = W^v \boldsymbol{x}_3$

$\boldsymbol{k}_4 = W^k \boldsymbol{x}_4$
$\boldsymbol{v}_4 = W^v \boldsymbol{x}_4$

Attention

Self-attention mechanism

Query: $\boldsymbol{q}$
Key: $\boldsymbol{k}$
Value: $\boldsymbol{v}$



$\boldsymbol{q}_1 = W^q\boldsymbol{x}_1$
$\boldsymbol{k}_1 = W^k\boldsymbol{x}_1$
$\boldsymbol{v}_1 = W^v\boldsymbol{x}_1$

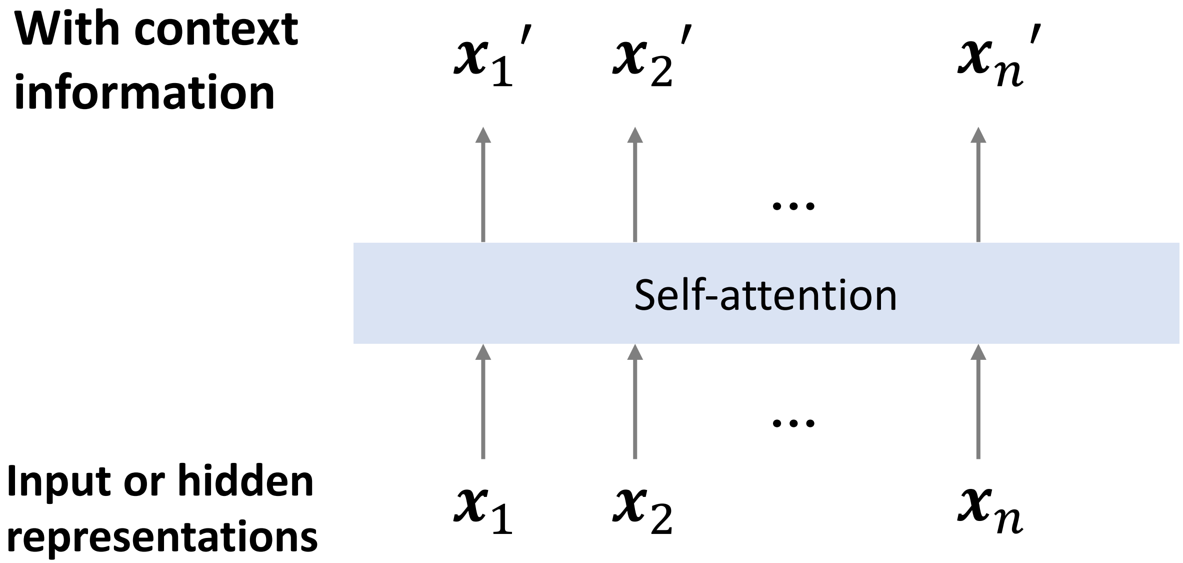
$\boldsymbol{k}_2 = W^k\boldsymbol{x}_2$
$\boldsymbol{v}_2 = W^v\boldsymbol{x}_2$

$\boldsymbol{k}_3 = W^k\boldsymbol{x}_3$
$\boldsymbol{v}_3 = W^v\boldsymbol{x}_3$

$\boldsymbol{k}_4 = W^k\boldsymbol{x}_4$
$\boldsymbol{v}_4 = W^v\boldsymbol{x}_4$

Attention

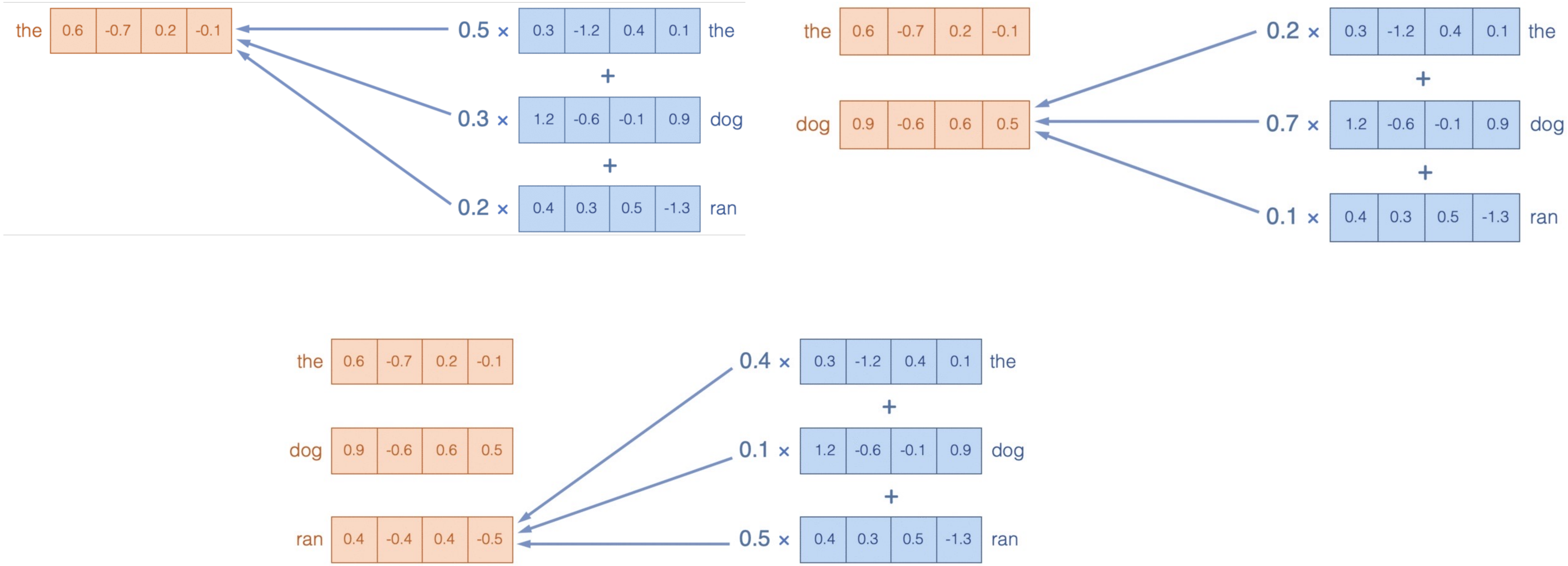
Self-attention mechanism



$$Attention(K, V, Q) = softmax\left(\frac{QK^T}{\sqrt{n}}\right)V$$

Attention

Composite embeddings based on attentions



the 0.6 -0.7 0.2 -0.1

0.5 × 0.3 -1.2 0.4 0.1 the

+

0.3 × 1.2 -0.6 -0.1 0.9 dog

+

0.2 × 0.4 0.3 0.5 -1.3 ran

the 0.6 -0.7 0.2 -0.1

dog 0.9 -0.6 0.6 0.5

0.2 × 0.3 -1.2 0.4 0.1 the

+

0.7 × 1.2 -0.6 -0.1 0.9 dog

+

0.1 × 0.4 0.3 0.5 -1.3 ran

the 0.6 -0.7 0.2 -0.1

dog 0.9 -0.6 0.6 0.5

ran 0.4 -0.4 0.4 -0.5

0.4 × 0.3 -1.2 0.4 0.1 the

+

0.1 × 1.2 -0.6 -0.1 0.9 dog

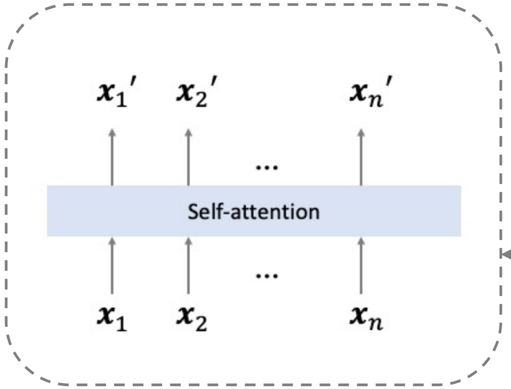
+

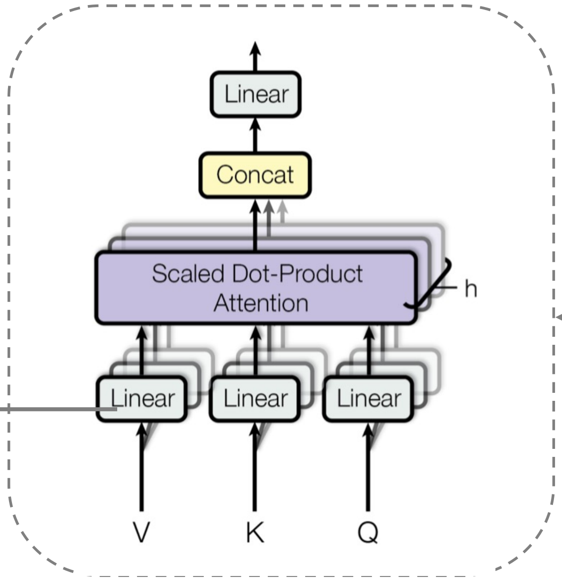
0.5 × 0.4 0.3 0.5 -1.3 ran

Source: https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1

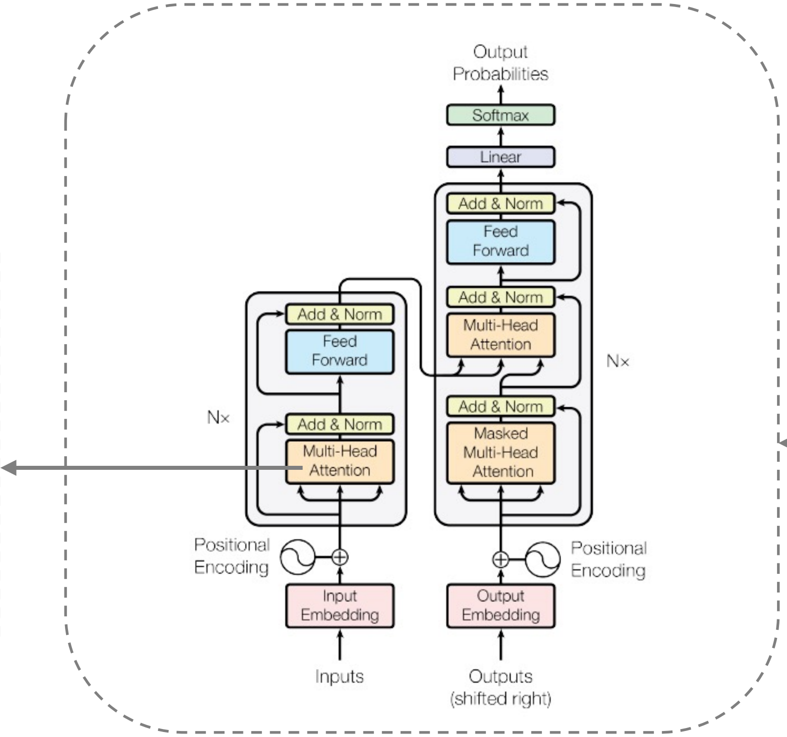
Attention

Consider the last attention layer for model interpretation

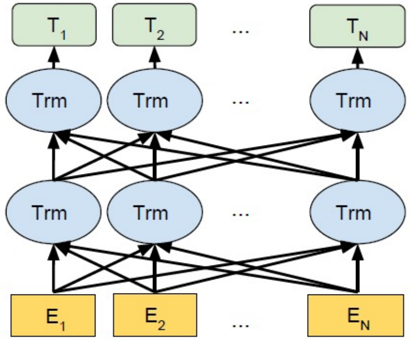




Multi-Head Attention



Transformer



BERT

Question?

Is Attention Interpretable?

Sofia Serrano, Noah A. Smith

(ACL, 2019)

Attention for Explanation

Attention weights can be highly inconsistent with model prediction

Input	x_1	x_2	$\cdots$	x_n
Attention	a_1	a_2	$\cdots$	a_n

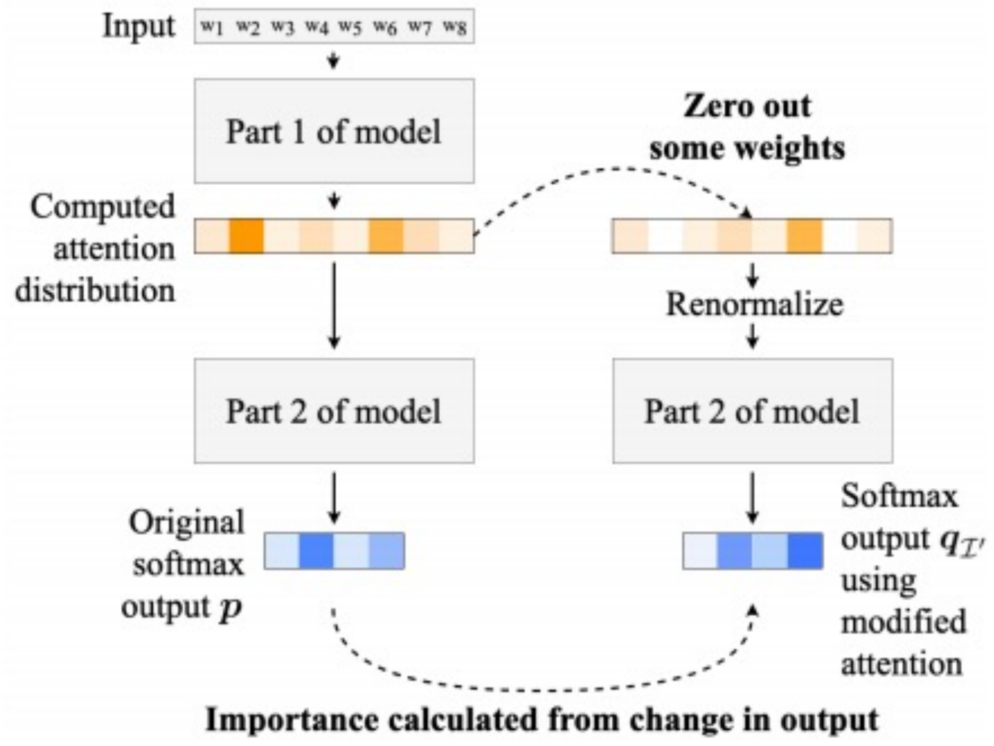
Sum to 1

Intermediate Representation Erasure

- Explanation I: a ranking of importance of the attention layer's input representations
- Exam the impact of some contextualized inputs to an attention layer, $I' \subset I$, on the model's output

Intermediate Representation Erasure

- Explanation I: a ranking of importance of the attention layer's input representations
- Exam the impact of some contextualized inputs to an attention layer, $I' \subset I$, on the model's output
- Running the model twice: once without any modification, once with the attention weights of I' zeroed out


Input
w1 w2 w3 w4 w5 w6 w7 w8
Part 1 of model
Computed attention distribution
Zero out some weights
Renormalize
Part 2 of model
Part 2 of model
Original softmax output p
Softmax output $q_{\mathcal{I}'}$ using modified attention
Importance calculated from change in output

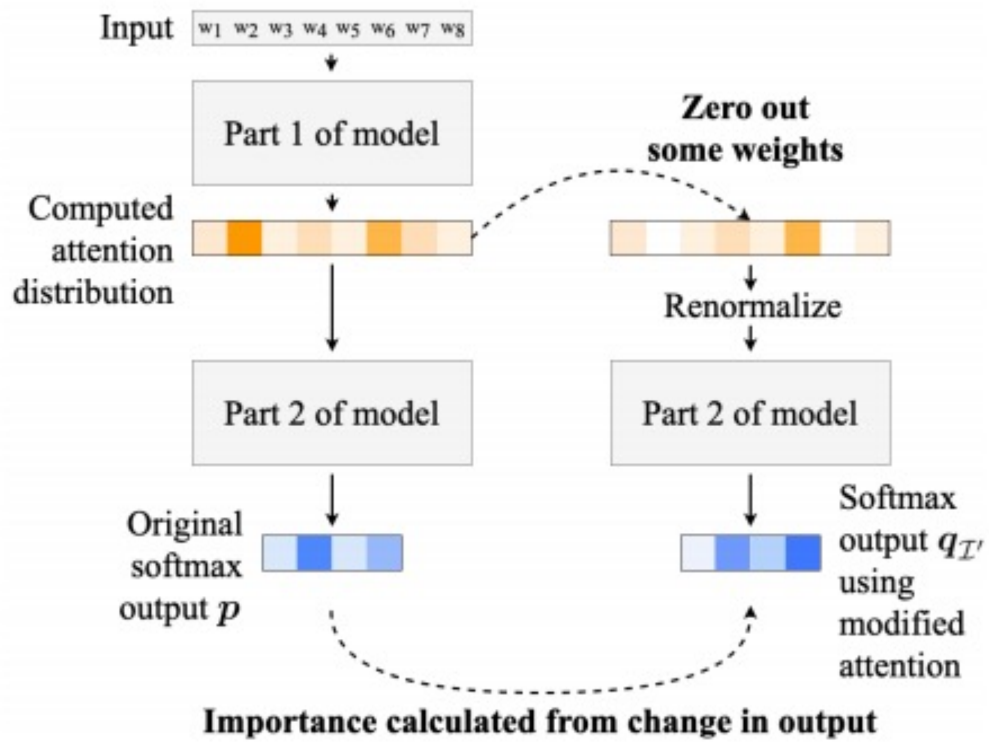
Intermediate Representation Erasure

Evaluate model prediction change

- Jensen-Shannon (JS) divergence between output distributions p and $q_{I'}$

$$JS(P|Q) = \frac{1}{2}KL(P|M) + \frac{1}{2}KL(Q|M)$$
$$M = \frac{1}{2}P + \frac{1}{2}Q$$

- Difference between the argmaxes of p and $q_{I'}$ (decision flip)


Input
w1 w2 w3 w4 w5 w6 w7 w8
Part 1 of model
Zero out some weights
Computed attention distribution
Renormalize
Part 2 of model
Part 2 of model
Original softmax output p
Softmax output $q_{\mathcal{I}'}$ using modified attention
Importance calculated from change in output

Single Attention Weight Importance

Remove the component $i^* \in I$ with the highest attention weight α_{i^*} $JS(p, q_{\{i^*\}})$

Comparison: a random component r drawn from I $JS(p, q_{\{r\}})$

Single Attention Weight Importance

Remove the component $i^* \in I$ with the highest attention weight α_{i^*} $JS(p, q_{\{i^*\}})$

Comparison: a random component r drawn from I $JS(p, q_{\{r\}})$

$$\nabla JS = JS(p, q_{\{i^*\}}) - JS(p, q_{\{r\}})$$

Indicate how important i^* is wrt r. Intuitively, if $\nabla\alpha = \alpha_{i^*} - \alpha_r$ is larger, ∇JS should be larger.

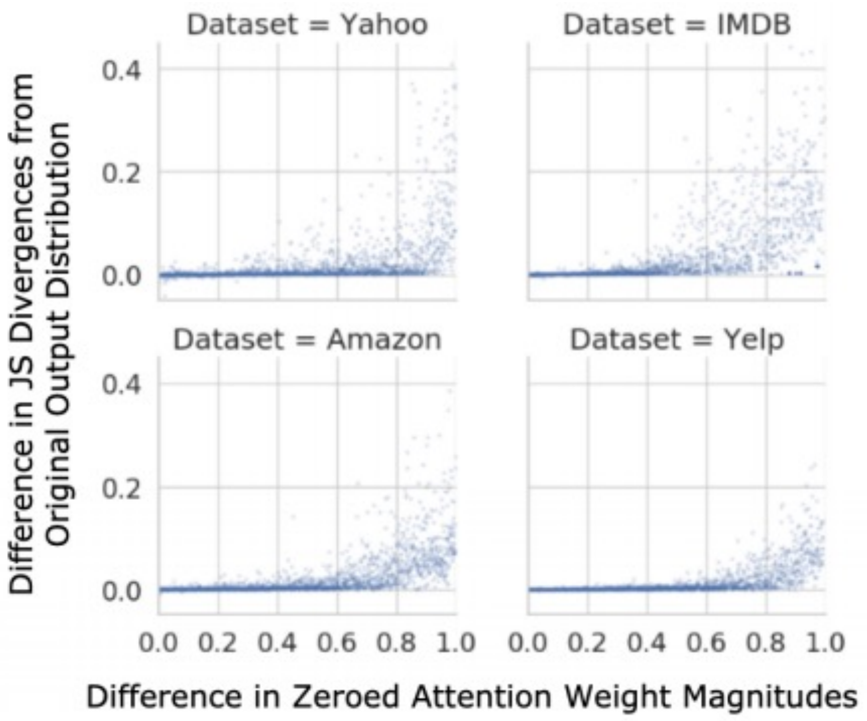
Single Attention Weight Importance

Remove the component $i^* \in I$ with the highest attention weight α_{i^*} $JS(p, q_{\{i^*\}})$

Comparison: a random component r drawn from I $JS(p, q_{\{r\}})$

$$\nabla JS = JS(p, q_{\{i^*\}}) - JS(p, q_{\{r\}})$$

∇JS



$$\nabla \alpha = \alpha_{i^*} - \alpha_r$$

- ✓ If i^* is more important, ∇JS is larger
- ✓ When ∇JS is small (close to 0), $\nabla \alpha$ tends to be small

  (i^* and r are nearly “tied” in attention)

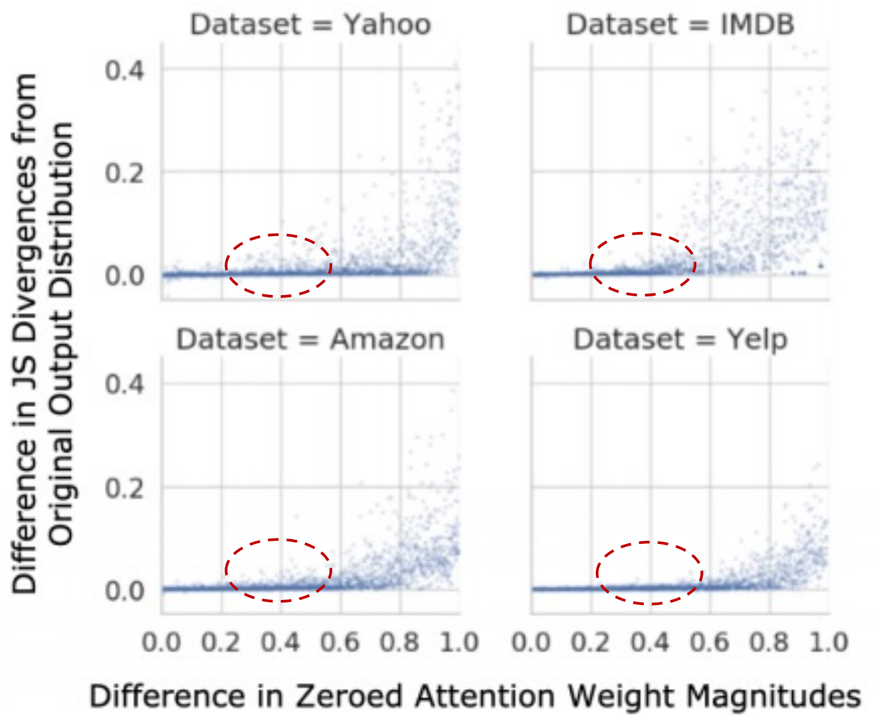
Single Attention Weight Importance

Remove the component $i^* \in I$ with the highest attention weight α_{i^*} $JS(p, q_{\{i^*\}})$

Comparison: a random component r drawn from I $JS(p, q_{\{r\}})$

$$\nabla JS = JS(p, q_{\{i^*\}}) - JS(p, q_{\{r\}})$$

∇JS



$$\nabla\alpha = \alpha_{i^*} - \alpha_r$$

- ✓ If i^* is more important, ∇JS is larger
- ✓ When ∇JS is small (close to 0), $\nabla\alpha$ tends to be small

  (i^* and r are nearly "tied" in attention)
- ✓ When $\nabla\alpha$ is about 0.4, ∇JS is still close to 0

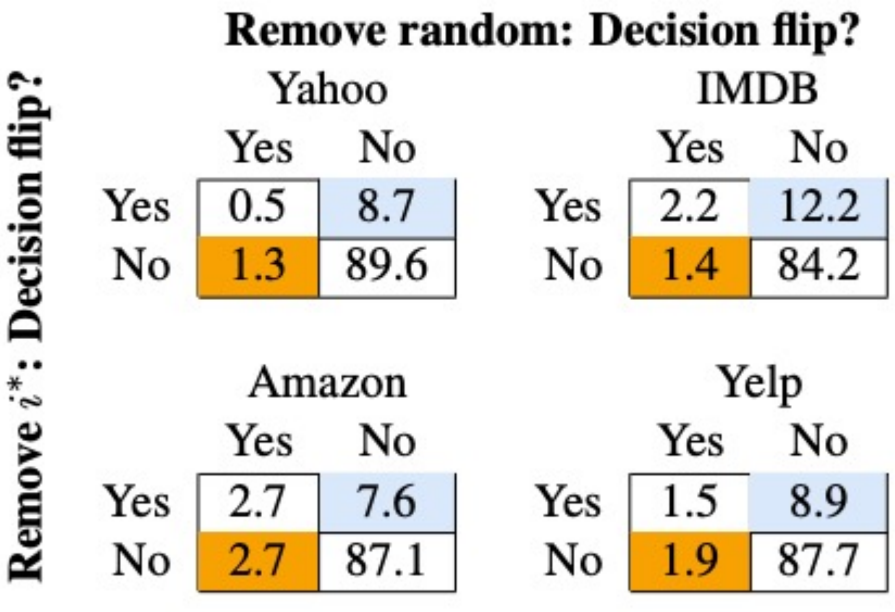
How much the attention weight can express the importance of a feature?

Single Attention Weight Importance

Decision flips caused by zeroing attention

Remove the component $i^* \in I$ with the highest attention weight α_{i^*}

Comparison: a random component r drawn from I



Remove random: Decision flip?

Remove i^*: Decision flip?	Yahoo		IMDB	
	Yes	No	Yes	No
Yes	0.5	8.7	2.2	12.2
No	1.3	89.6	1.4	84.2

Remove i^*: Decision flip?	Amazon		Yelp	
	Yes	No	Yes	No
Yes	2.7	7.6	1.5	8.9
No	2.7	87.1	1.9	87.7

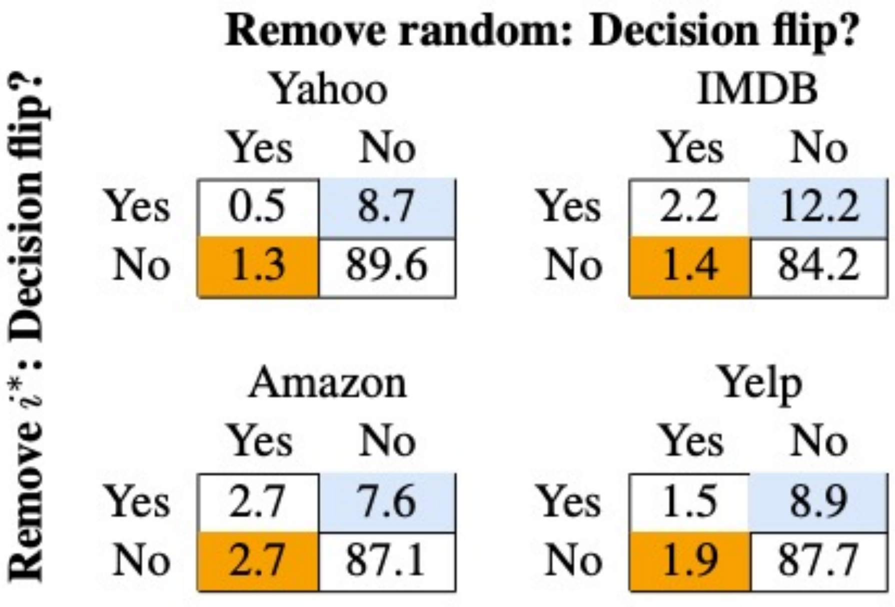
Intuitively, upper-right values should be much larger than lower-left values

Single Attention Weight Importance

Decision flips caused by zeroing attention

Remove the component $i^* \in I$ with the highest attention weight α_{i^*}

Comparison: a random component r drawn from I



Remove random: Decision flip?

Remove i^*: Decision flip?

	Yahoo			IMDB	
	Yes	No		Yes	No
Yes	0.5	8.7	Yes	2.2	12.2
No	1.3	89.6	No	1.4	84.2

	Amazon			Yelp	
	Yes	No		Yes	No
Yes	2.7	7.6	Yes	1.5	8.9
No	2.7	87.1	No	1.9	87.7

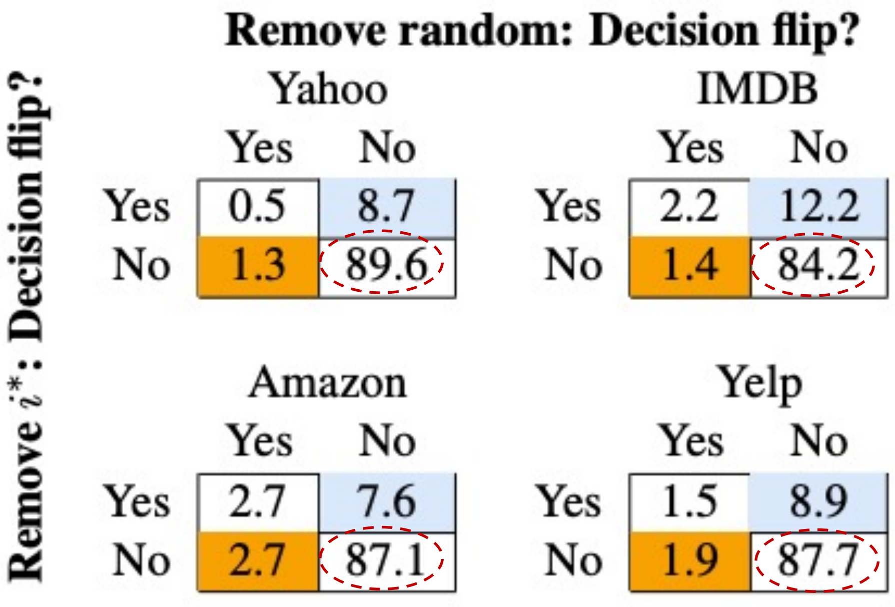
✓ Upper-right values are larger than lower-left values (removing i^* is easier to flip decision)

Single Attention Weight Importance

Decision flips caused by zeroing attention

Remove the component $i^* \in I$ with the highest attention weight α_{i^*}

Comparison: a random component r drawn from I



Remove random: Decision flip?

Remove i^*: Decision flip?

Yahoo	Yes	No
Yes	0.5	8.7
No	1.3	89.6

IMDB	Yes	No
Yes	2.2	12.2
No	1.4	84.2

Amazon	Yes	No
Yes	2.7	7.6
No	2.7	87.1

Yelp	Yes	No
Yes	1.5	8.9
No	1.9	87.7

- ✓ Upper-right values are larger than lower-left values (removing i^* is easier to flip decision)
- ✓ In most cases (lower-right values), erasing i^* does not change the decision

The highest attention weight indicates the most important feature?

Single Attention Weight Importance

$I \quad \boxed{\alpha_1} \quad \alpha_2 \quad \alpha_3 \quad \alpha_4 \quad \cdots \quad \alpha_n$ (descending order of importance)

Single Attention Weight Importance

$I \quad \boxed{\alpha_1} \quad \alpha_2 \quad \alpha_3 \quad \alpha_4 \quad \cdots \quad \alpha_n$ (descending order of importance)

$$\Downarrow$$

$I \quad \boxed{\alpha_1 \quad \alpha_2 \quad \alpha_3} \quad \alpha_4 \quad \cdots \quad \alpha_n$ (descending order of importance)

Intuitively, the top items in a truly useful ranking of importance would comprise a minimal necessary set of information for making the model's decision

Importance of Sets of Attention Weights

Test how multiple attention weights perform together as importance predictors

Erasing representations from the top of the ranking downward until the model’s decision changes

$I \quad \cancel{\alpha_1} \quad \cancel{\alpha_2} \quad \cancel{\alpha_3} \quad \alpha_4 \quad \cdots \quad \alpha_n$ → Prediction change

Importance of Sets of Attention Weights

Test how multiple attention weights perform together as importance predictors

Erasing representations from the top of the ranking downward until the model’s decision changes

$$I \quad \cancel{\alpha_1} \quad \cancel{\alpha_2} \quad \cancel{\alpha_3} \quad \alpha_4 \quad \cdots \quad \alpha_n \quad \Rightarrow \quad \text{Prediction change}$$

$$I_1 \quad \cancel{\alpha_1} \quad \cancel{\alpha_2} \quad \alpha_3 \quad \alpha_4 \quad \cdots \quad \alpha_n \quad \Rightarrow \quad \text{Prediction change}$$

(Alternative rankings
of importance)

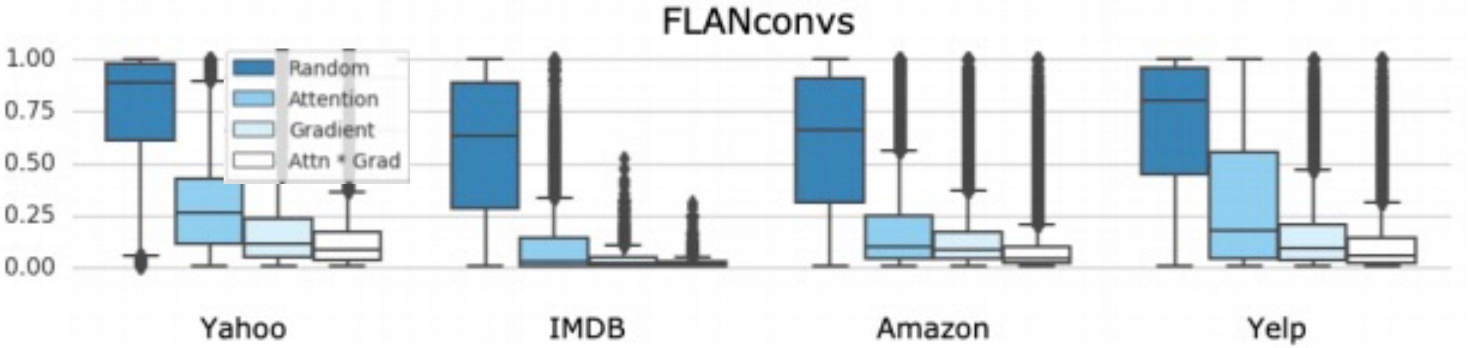
Attention may not be a good interpretation method

Importance of Sets of Attention Weights

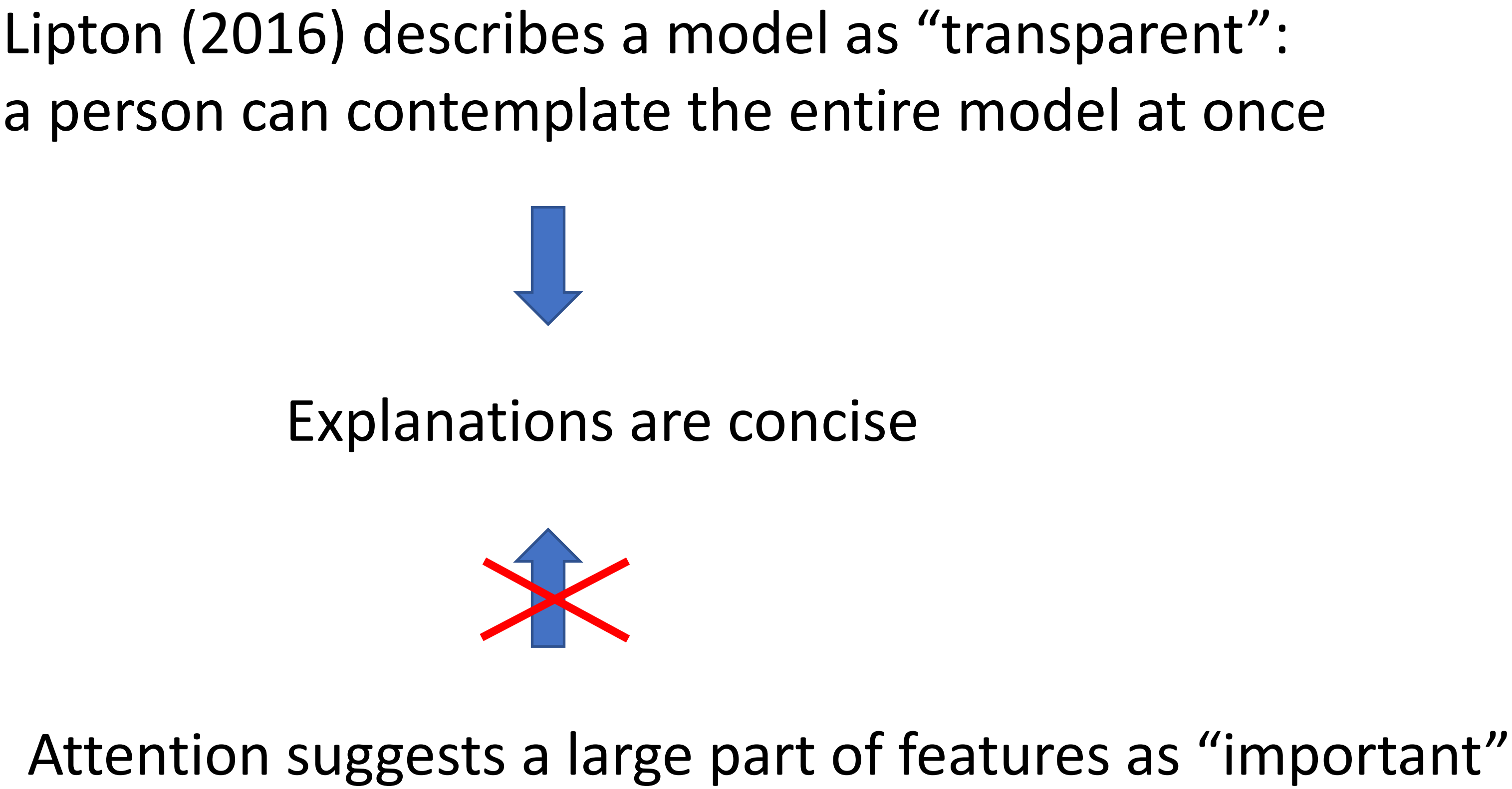
Baselines

- Random rankings
- Gradients
- Gradients × Attentions

Fractions of original components removed before first decision flip under different importance rankings



✓ Both a high attention weight and a high calculated gradient indicate an important component



Lipton (2016) describes a model as “transparent”:
a person can contemplate the entire model at once

Explanations are concise

Attention suggests a large part of features as “important”

Question?

Reference

- Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences." *International conference on machine learning*. PMLR, 2017.
- Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." *International conference on machine learning*. PMLR, 2017.
- Serrano, Sofia, and Noah A. Smith. "Is attention interpretable?." *arXiv preprint arXiv:1906.03731* (2019).
- Smilkov, Daniel, et al. "Smoothgrad: removing noise by adding noise." *arXiv preprint arXiv:1706.03825* (2017).
- Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
- Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).