Lecture 22: Background for "Attention is All You Need"

Mark Hasegawa-Johnson

ECE 537, Fall 2022

1. The Cauchy-Schwartz Inequality and Cosine Distance
2. Smart PCA
3. Encoder-Decoder Sequence-to-Sequence Models
4. Attention
5. Intra-Attention (Self-Attention) vs. Inter-Attention
6. Summary

Outline

1. The Cauchy-Schwartz Inequality and Cosine Distance
2. Smart PCA
3. Encoder-Decoder Sequence-to-Sequence Models
4. Attention
5. Intra-Attention (Self-Attention) vs. Inter-Attention
6. Summary

The Cauchy-Schwartz Inequality

The Cauchy-Schwart inequality says that, for any two vectors $\vec{x} = [x_1, \ldots, x_N]^T$ and $\vec{y} = [y_1, \ldots, y_N]^T$,

$$|\vec{x}^T \vec{y}| \leq \|\vec{x}\|^2 \|\vec{y}\|^2$$

If we define the unit vectors as follows,

$$\hat{x} = \frac{\vec{x}}{\|\vec{x}\|}, \quad \hat{y} = \frac{\vec{y}}{\|\vec{y}\|},$$

then the Cauchy-Schwartz inequality says that

$$-1 \leq \hat{x}^T \hat{y} \leq 1$$

The Cauchy-Schwartz Inequality: Proof

Suppose we have a particular $\vec{x}$, and we want to:

- choose $y_1, \ldots, y_N$ in order to maximize/minimize the dot product,
$$\vec{x}^T\vec{y} = \sum_{i=1}^{N} x_i y_i$$
- subject to the constraint that the length of $\vec{y}$ is fixed, say,
$$L^2 = \sum_{i=1}^{N} y_i^2$$

This can be done by choosing $y_1, \ldots, y_N$ to maximize/minimize the following Lagrangian:

$$\mathcal{L}(\vec{y}) = \sum_{i=1}^{N} x_i y_i - \frac{\kappa}{2}\left(\sum_{i=1}^{N} y_i^2 - L^2\right)$$

The Cauchy-Schwartz Inequality: Proof

$$\mathcal{L}(\vec{y}) = \sum_{i=1}^{N} x_i y_i - \frac{\kappa}{2}\left(\sum_{i=1}^{N} y_i^2 - L^2\right)$$

Setting $\frac{\partial \mathcal{L}}{\partial y_i} = 0$ yields:

$$y_i = \frac{x_i}{\kappa},$$

There are two values of κ that satisfy the constraint, and they give the maximizer/minimizer of the dot product, respectively:

$$\kappa = \pm\frac{\|\vec{x}\|}{\|\vec{y}\|}$$

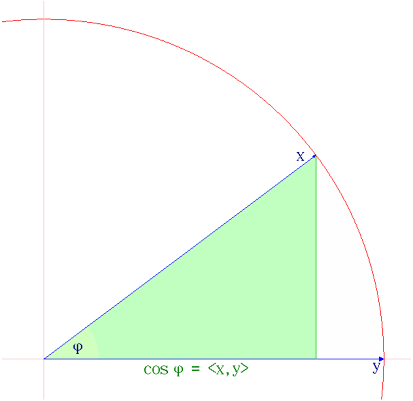
Cosine Distance

The Cauchy-Schwartz inequality can be written as:

$$-1 \leq \hat{x}^T \hat{y} \leq 1,$$

where $\hat{x} = \frac{\vec{x}}{\|\vec{x}\|}$ and $\hat{y} = \frac{\vec{y}}{\|\vec{y}\|}$. This is an N-dimensional generalization of the 2D geometric interpretation of the dot product:

$$\hat{x}^T \hat{y} = \cos\phi$$

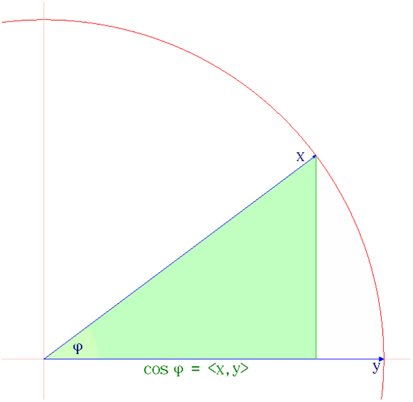


CC-SA 4.0, `https://commons.wikimedia.org/wiki/File:Cauchy-Schwarz_inequation_in_Euclidean_plane.gif`

Cosine Distance

Large-magnitude vectors have a tendency to swamp the training criterion for a neural net. It's often useful to explicitly ignore the magnitude of the vector, and to only measure the angle between two vectors on the $(N-1)$-dimensional hypersphere. This is done using the **cosine distance**,

$$\begin{aligned} \text{cosd}(\vec{x}, \vec{y}) &= 1 - \cos(\vec{x}, \vec{y}) \\ &= 1 - \hat{x}^T \hat{y} \end{aligned}$$



CC-SA 4.0,

https://commons.wikimedia.org/wiki/File:Cauchy-Schwarz_inequation_in_Euclidean_plane.gif

Outline

1. The Cauchy-Schwartz Inequality and Cosine Distance
2. Smart PCA
3. Encoder-Decoder Sequence-to-Sequence Models
4. Attention
5. Intra-Attention (Self-Attention) vs. Inter-Attention
6. Summary

PCA and Smart PCA

Consider trying to find a set of vectors, $\vec{w}_k$ $(1 \leq k \leq K)$, in order to minimize

$$\text{MSE} = E\left[\|\vec{x} - \sum_{k=1}^{K} \rho_k \vec{w}_k\|^2\right],$$

where expectation is over the training dataset. The minimum-MSE solution has orthogonal vectors, and therefore $\rho_k = \frac{\vec{w}_k^T \vec{x}}{\|\vec{w}_k\|^2}$ is the minimum-MSE weight.

- The MMSE solution can be computed using principal components analysis (PCA).
- The MMSE solution can also be computed using an autoencoder neural network. If K is much less than the vector dimension, the autoencoder computation is faster, so this is called "smart PCA" (SPCA).

Consider a training database, $\{\vec{x}_1, \ldots, \vec{x}_n\}$. An autoencoder computes

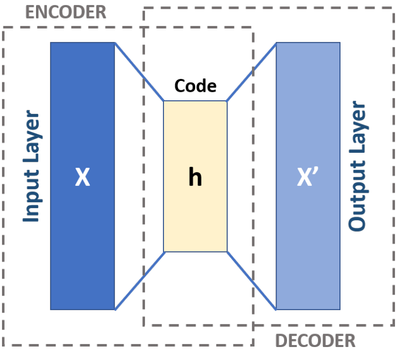
$$h_k^i = \vec{x}_i^T \vec{w}_k,$$

and

$$\vec{x}_i' = \sum_{k=1}^{K} h_k^i \vec{w}_k,$$

then trains $\vec{w}_k$ to minimize

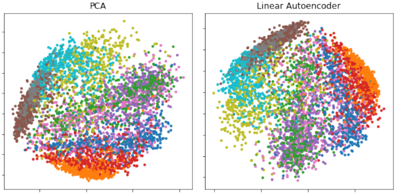
$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \|\vec{x}_i - \vec{x}_i'\|^2$$



Both PCA and SPCA choose unit-length vectors, $\vec{w}_k$ in order to minimize

$$\text{MSE} = E\left[\|\vec{x} - \sum_{k=1}^{K} \rho_k \vec{w}_k\|^2\right],$$

therefore both SPCA and PCA choose vectors that span the same vector subspace. SPCA does not decide the order of the vectors, or their sign, so SPCA can produce a vector subspace that's a rotated version of the one chosen by PCA.



Plot of the first two Principal Components (left) and the two hidden units' values of a Linear Autoencoder (right) applied to the Fashion MNIST dataset. The two models being both linear learn to span the same subspace.

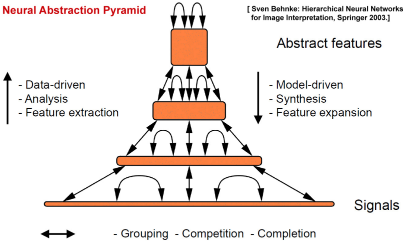
PCA and Smart PCA

The training criterion for both PCA and SPCA is

$$\begin{aligned}
\text{MSE} &= E\left[\|\vec{x} - \sum_{k=1}^{K} \rho_k \vec{w}_k\|^2\right], \quad \rho_k = \frac{\vec{w}_k^T \vec{x}}{\|\vec{w}_k\|^2} \\
&= E\left[\|\vec{x}\|^2 - \sum_{k=1}^{K} \frac{(\vec{w}_k^T \vec{x})^2}{\|\vec{w}_k\|^2}\right] \\
&= E\left[\|\vec{x}\|^2 \left(1 - \sum_{k=1}^{K} \left(\hat{w}_k^T \hat{x}\right)^2\right)\right] \\
&= E\left[\|\vec{x}\|^2 \left(1 - \sum_{k=1}^{K} \cos^2\left(\vec{w}_k, \vec{x}\right)\right)\right]
\end{aligned}$$

So the MSE minimizer is a set of vectors with approximately the minimum average squared cosine distance to the data.

- Since LeCun's 1991 paper, most image CNNs are basically autoencoders, in their lower layers. The filters at lower layers learn principal components that best match the local image patches.
- As suggested by this figure, the same convolution kernels can also be used to resynthesize the image, using an autoencoder training criterion.



CC-SA 4.0,
`https://commons.wikimedia.org/wiki/File:Neural_Abstraction_Pyramid.jpg`

Outline

1. The Cauchy-Schwartz Inequality and Cosine Distance
2. Smart PCA
3. Encoder-Decoder Sequence-to-Sequence Models
4. Attention
5. Intra-Attention (Self-Attention) vs. Inter-Attention
6. Summary

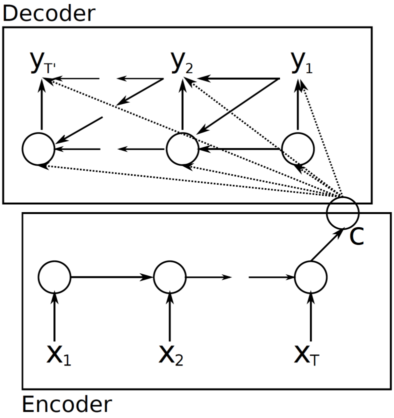
ASR and Machine Translation (MT): Similarities and Differences

	Input, Output Lengths	Input, Output Sequence Order
ASR	Different	Same
MT	Different	Different

Encoder-Decoder Neural Nets

Cho et al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation" (Sep. 2014) proposed a solution:

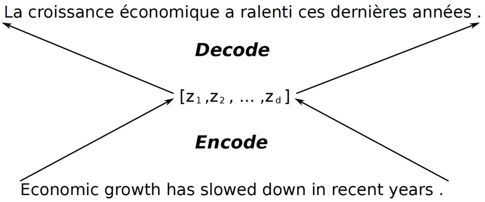
- RNN "encodes" all of the input to a summary vector, C, then
- RNN "decodes" C in order to produce the output.



Cho, Merriënboer, Gulcehre, Bahdenau, Bougares, Schwenk & Bengio, Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, Fig. 1.

Encoder-Decoder Neural Nets

Cho et al., "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches" (Oct. 2014) proposed that the encoder summary didn't need to be a single vector, it could be a sequence of vectors.



Cho, Merriënboer, Bahdenau & Bengio, On the Properties of Neural Machine Translation: Encoder–Decoder Approaches, Fig. 3.

Outline

1. The Cauchy-Schwartz Inequality and Cosine Distance
2. Smart PCA
3. Encoder-Decoder Sequence-to-Sequence Models
4. Attention
5. Intra-Attention (Self-Attention) vs. Inter-Attention
6. Summary

Encoder-Decoder: Too Much Information?

- Encoder-decoder models beat the state of the art in some tasks (usually tasks with a lot of data), but had a fatal flaw.
- If the encoder creates a small summary, then it accidentally throws away important information, because it doesn't always know which information is important.
- If the encoder creates a large summary, then the decoder doesn't know which data to use for any given computation, so training takes longer, and sometimes fails.

Attention

In December 2014, Chorowski et al. proposed a new type of encoder-decoder algorithm, based on "attention."

- The i^{th} time step in the **Decoder** is computed based on an attention-weighted summary of the inputs:

$$s_i = \text{Recurrency}\left(s_{i-1}, \sum_{j=1}^{L} \alpha_{i,j} h_j\right)$$

- The **Attention** is a kind of probability mass (sums to one) over the different input time steps, $1 \le j \le L$:

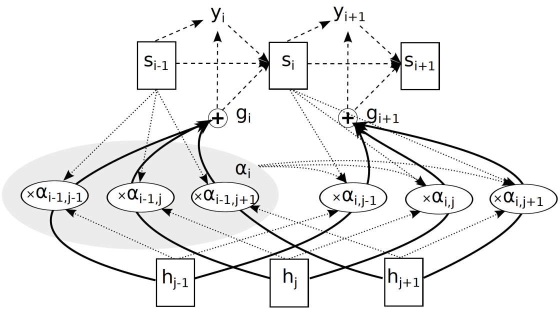
$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{j=1}^{L} \exp(e_{i,j})}, \quad \sum_{j=1}^{L} \alpha_{i,j} = 1$$

- The **Attention Score** is a special neural net that measures the similarity between input vector h_j and output vector s_{i-1}:

$$e_{i,j} = \text{Score}(s_{i-1}, h_j)$$

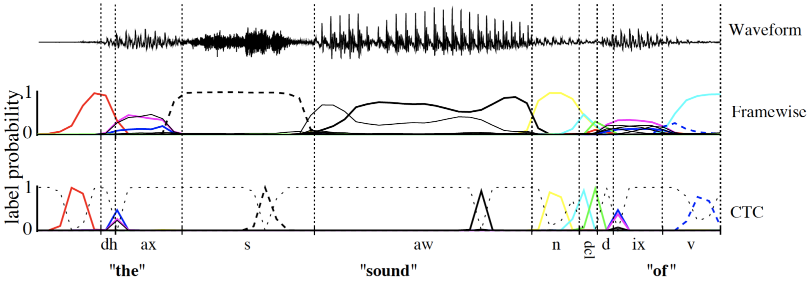
Attention

- s_{i-1} and h_j determine $\alpha_{i,j}$
- s_i is determined by $\sum_{j=1}^{L} \alpha_{i,j} h_j$



Chorowski, Bahdanau, Serdyk, Cho & Bengio, Attention-Based Models for Speech Recognition, Fig. 1

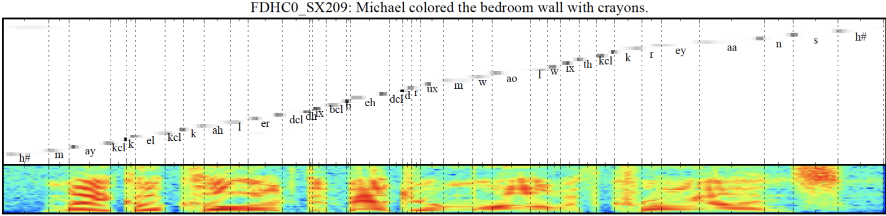
Segmentation: The CTC Viewpoint



In CTC, **time alignment** means finding input time points (t) where you can align each of the output labels (s).

Graves et al., 2006, Figure 1. (c) ICML

Segmentation: The Attention Viewpoint



In attention-based models, **time alignment** means computing $\alpha_{i,j}$ as a function of i (vertical axis) and j (horizontal axis).

Chorowski et al., 2014, Figure 3.

Outline

1. The Cauchy-Schwartz Inequality and Cosine Distance
2. Smart PCA
3. Encoder-Decoder Sequence-to-Sequence Models
4. Attention
5. Intra-Attention (Self-Attention) vs. Inter-Attention
6. Summary

When should you pay attention?

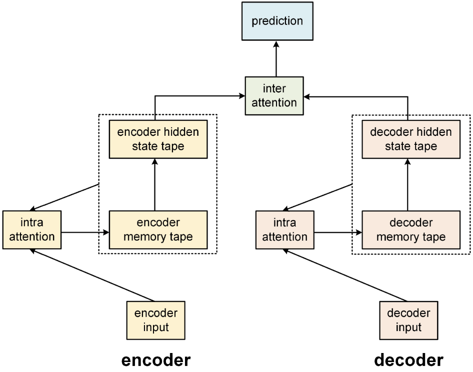
- The key idea of attention is that there is some sort of similarity score, $e_{i,j} = \text{Similarity}(s_{i-1}, h_j)$, so that you can compute attention weights according to

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{j=1}^{L} \exp(e_{i,j})}$$

- Raw inputs (speech) and raw outputs (text) are not inherently similar.
- There needs to be a network that converts the inputs into some hidden vectors, h_j and s_{i-1}, that are similar if and only if $\alpha_{i,j}$ should be large.

Cheng, Dong and Lapata, "Long Short-Term Memory-Networks for Machine Reading," proposed using a new kind of attention called "intra-attention" or "self-attention" to compute

- h_j from the inputs, and
- s_{i-1} from the preceding outputs, so that
- inter-attention will correctly measure Similarity(s_{i-1}, h_j).



Cheng, Dong & Lapata, "Long Short-Term Memory-Networks for Machine Reading," 2016, Fig. 3(a)

Intra-Attention

An RNN with intra-attention computes the RNN state vector $\tilde{h}_t$ as the attention-weighted summary of past RNN state vectors:

$$\tilde{h}_t = \sum_{i=1}^{t-1} \alpha_{t,i} h_i,$$

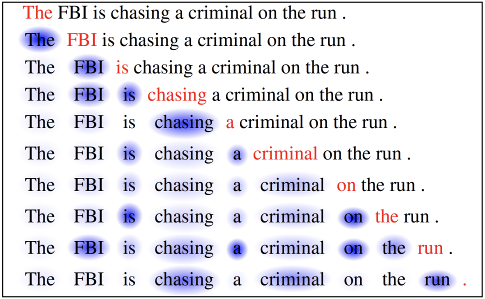
where the weights, $\alpha_{t,i}$, are softmax normalized:

$$\alpha_{t,i} = \text{softmax}(a_{t,i}),$$

based on similarities computed between h_i, $\tilde{h}_{t-1}$, and the input vector x_t:

$$a_{t,i} = v^T \tanh\left(W_h h_i + W_x x_t + W_{\tilde{h}} \tilde{h}_{t-1}\right)$$

- The representation of each word (in red) is computed based on. . .
- an attention-weighted summary of all previous words (attention weights in blue).
- Thus, the meaning of a word depends on its context.



Cheng, Dong & Lapata, "Long Short-Term Memory-Networks for Machine Reading," 2016, Fig. 1

Outline

1. The Cauchy-Schwartz Inequality and Cosine Distance
2. Smart PCA
3. Encoder-Decoder Sequence-to-Sequence Models
4. Attention
5. Intra-Attention (Self-Attention) vs. Inter-Attention
6. Summary

Summary

- PCA or smart PCA (autoencoder) learns a set of vectors with maximum cumulative similarity to the input:

$$\text{MSE} = E\left[\|\vec{x}\|^2\left(1-\sum_{k=1}^{K}\cos^2\left(\vec{w}_k,\vec{x}\right)\right)\right]$$

- An encoder-decoder network can do something CTC can't: re-arrange the order.
- Attention:

$$s_i = \text{Recurrency}\left(s_{i-1},\sum_{j=1}^{L}\alpha_{i,j}h_j\right)$$

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{j=1}^{L}\exp(e_{i,j})}$$

$$e_{i,j} = \text{Similarity}(s_{i-1},h_j)$$