Learning Efficient Representations for Sequence Retrieval

Colin Raffel
Boston Data Festival
September 19, 2015
Sequence Retrieval
Dynamic Time Warping
Dynamic Time Warping
Making DTW work

1. Compute a pairwise distance matrix of sequences
Making DTW work

1. Compute a pairwise distance matrix of sequences
2. Use DTW to find lowest-cost path through the distance matrix
Making DTW work

1. Compute a pairwise distance matrix of sequences
2. Use DTW to find lowest-cost path through the distance matrix
3. Allow subsequence matching, with some tolerance
Making DTW work

1. Compute a pairwise distance matrix of sequences
2. Use DTW to find lowest-cost path through the distance matrix
3. Allow subsequence matching, with some tolerance
4. Use an additive penalty (e.g. median distance)
Making DTW work

1. Compute a pairwise distance matrix of sequences
2. Use DTW to find lowest-cost path through the distance matrix
3. Allow subsequence matching, with some tolerance
4. Use an additive penalty (e.g. median distance)
5. Compute the total distance between aligned frames
Making DTW work

1. Compute a pairwise distance matrix of sequences
2. Use DTW to find lowest-cost path through the distance matrix
3. Allow subsequence matching, with some tolerance
4. Use an additive penalty (e.g. median distance)
5. Compute the total distance between aligned frames
6. Normalize by path length and mean of path submatrix
DTW Issues

- $O(NM)$-complex using dynamic programming

Various "pruning methods" exist which approach linear time...

However, most are not universally applicable.

Data dimensionality can cause expensive "local distance" calculations.

Quadratic penalty when the data is sampled too finely.

Inappropriate when sequences come from different modalities.

Relies on a non-learned metric for comparing feature vectors.
DTW Issues

- $O(NM)$-complex using dynamic programming
- Various “pruning methods” exist which approach linear time...
DTW Issues

- $O(NM)$-complex using dynamic programming
- Various “pruning methods” exist which approach linear time...
- However, most are not universally applicable
DTW Issues

- $O(NM)$-complex using dynamic programming
- Various “pruning methods” exist which approach linear time...
- However, most are not universally applicable
- Data dimensionality can cause expensive “local distance” calculations
DTW Issues

- $O(NM)$-complex using dynamic programming
- Various “pruning methods” exist which approach linear time...
- However, most are not universally applicable
- Data dimensionality can cause expensive “local distance” calculations
- Quadratic penalty when the data is sampled too finely
DTW Issues

- $O(NM)$-complex using dynamic programming
- Various “pruning methods” exist which approach linear time...
- However, most are not universally applicable
- Data dimensionality can cause expensive “local distance” calculations
- Quadratic penalty when the data is sampled too finely
- Inappropriate when sequences come from different modalities
DTW Issues

- $O(NM)$-complex using dynamic programming
- Various “pruning methods” exist which approach linear time...
- However, most are not universally applicable
- Data dimensionality can cause expensive “local distance” calculations
- Quadratic penalty when the data is sampled too finely
- Inappropriate when sequences come from different modalities
- Relies on a non-learned metric for comparing feature vectors
Similarity-Preserving Hashing
Similarity-Preserving Hashing
Hash Sequences

distance[$m, n$] = bits_set[$x[m] \oplus y[n]$]
Loss function
Training details

- Training data is pairs of feature vectors from successfully pre-aligned sequences
Training details

- Training data is pairs of feature vectors from successfully pre-aligned sequences
- Negative examples chosen at random
Training details

- Training data is pairs of feature vectors from successfully pre-aligned sequences
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
Training details

- Training data is pairs of feature vectors from successfully pre-aligned sequences
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
Training details

- Training data is pairs of feature vectors from successfully pre-aligned sequences
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
- Optimization using RMSProp

No other regularization needed

Hyperparameters chosen using Whetlab (RIP)

Objective: Bhattacharyya distance of positive/negative examples distance distributions
Training details

- Training data is pairs of feature vectors from successfully pre-aligned sequences
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
- Optimization using RMSProp
- No other regularization needed
Training details

- Training data is pairs of feature vectors from successfully pre-aligned sequences
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
- Optimization using RMSProp
- No other regularization needed
- Hyperparameters chosen using Whetlab (RIP)
Training details

- Training data is pairs of feature vectors from successfully pre-aligned sequences
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
- Optimization using RMSProp
- No other regularization needed
- Hyperparameters chosen using Whetlab (RIP)
- Objective: Bhattacharyya distance of positive/negative examples distance distributions
Network Structure

- Two different networks with the same structure used for sequences in each modality
Network Structure

- Two different networks with the same structure used for sequences in each modality
- Two convolutional layers: 5x12 and 3x3
Network Structure

- Two different networks with the same structure used for sequences in each modality
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
Network Structure

- Two different networks with the same structure used for sequences in each modality
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each

ReLUs throughout, with tanh on the output
16-bit hashes created by thresholding output
Weight matrices initialized using He’s method, $\{fan_{in}\}$
Bias vectors all initialized to zero
Network made out of lasagne
Network Structure

- Two different networks with the same structure used for sequences in each modality
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
Network Structure

- Two different networks with the same structure used for sequences in each modality
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
- 16-bit hashes created by thresholding output

Weight matrices initialized using He’s method, a

Bias vectors all initialized to zero

Network made out of lasagne
Network Structure

- Two different networks with the same structure used for sequences in each modality
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
- 16-bit hashes created by thresholding output
- Weight matrices initialized using He’s method, $\sqrt{2/fan\_in}$
Network Structure

- Two different networks with the same structure used for sequences in each modality
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
- 16-bit hashes created by thresholding output
- Weight matrices initialized using He’s method, $\sqrt{2/fan\_in}$
- Bias vectors all initialized to zero
Network Structure

- Two different networks with the same structure used for sequences in each modality
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
- 16-bit hashes created by thresholding output
- Weight matrices initialized using He’s method, $\sqrt{2/fan\_in}$
- Bias vectors all initialized to zero
- Network made out of lasagne
Validation Distance Distribution

![Graph showing a bar chart with two categories: Similar and Dissimilar. The x-axis represents Distance, ranging from 0 to 18, and the y-axis represents Proportion, ranging from 0.00 to 0.30. The bars indicate the proportion of similar and dissimilar data points at various distances.](image)
Example Sequence

7 digital audio CQT

Synthesized MIDI CQT

Audio hash sequence

MIDI hash sequence

CQT distance matrix

Hash sequence Hamming distance matrix
First Layer Filters
Correct Match Rank Results

![Graph showing Correct Match Rank Results]

- **Hash Sequence DTW**
- **Baseline**
Sequence Embedding
Sentence Embeddings, with t-SNE

Sutskever et. al; “Sequence to Sequence Learning with Neural Networks”
Sequence Embedding
Sequence Embedding
Attention

\[ \alpha = \text{softmax}(wx + b) \]

\[ w \in \mathbb{R}^{n\text{-features}}, \ b \in \mathbb{R}, \ \alpha \in \mathbb{R}^{n\text{-steps}} \]
Other Differences

- Batches of entire (cropped) sequences
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not pre-aligned
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not pre-aligned
- Re-tune hyperparameters with `simple_spearmint`
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not pre-aligned
- Re-tune hyperparameters with `simple_spearmint`
- Only use 1 convolution/max pooling layer
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not pre-aligned
- Re-tune hyperparameters with `simple_spearmint`
- Only use 1 convolution/max pooling layer
- Add an attention layer between convolution and dense
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not pre-aligned
- Re-tune hyperparameters with simple_spearmint
- Only use 1 convolution/max pooling layer
- Add an attention layer between convolution and dense
- Output is now $[-1, 1]^{128}$
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not pre-aligned
- Re-tune hyperparameters with simple_spearmint
- Only use 1 convolution/max pooling layer
- Add an attention layer between convolution and dense
- Output is now $[-1, 1]^{128}$
- Network structure is otherwise the same
Validation Distance Distribution

Similar
Dissimilar

Proportion
Distance

0 0.0 0.1 0.2 0.3 0.4 0.5
0 5 10 15 20

Distance
Example Embeddings
Embedding Distance Matrix
Correct Match Rank Results

![Graph showing Correct Match Rank Results with two lines: Embedding and Baseline. The x-axis represents Rank on a log scale from $10^0$ to $10^5$, and the y-axis represents Percentage Below on a linear scale from 0 to 100. The Embedding line starts close to the x-axis and increases steeply to the top, while the Baseline line starts slightly above the x-axis and increases more gradually.]
Thanks!

craffel@gmail.com

http://github.com/craffel/