Zurich University
of Applied Sciences

Life Sciences and
Facility Management

Learning Long-term Dependencies in Recurrent Neural Networks

Stefan Glüge

ZHAW: Zurich University of Applied Sciences - Institute of Applied Simulation - Predictive & Bio-inspired Modelling

Table of contents

Sequence Classification and Prediction

- *Learn class label* corresponding to a given sequence of input vectors

$$\begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}_t \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}_{t+1} \cdots \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}_{t=T} \Longrightarrow C \tag{1}$$

- *Predict next element* of a given sequence of input vectors

$$\begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}_t \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}_{t+1} \cdots \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}_{t=T} \Longrightarrow \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}_{t=T+1} \tag{2}$$

Application of Recurrent Networks in Sequence Learning

- Sequence Classification
  - Classification of EEG signals (Forney & Anderson, 2011)
  - Visual pattern recognition: handwritten char. (Nishide et al., 2011)
  - Seismic signal classification (Park et al., 2011)
  - Pattern recognition in images (Abou-Nasr, 2010)
  - Emotion recognition from speech (Glüge et al., 2011)

- Sequence Prediction
  - Load forecasting in electric power systems (Barbounis et al., 2006)
  - Automatic speech processing (Varoglu & Hacioglu, 1999)
  - Sunspot series prediction (Park, 2011)
  - Network traffic prediction (Bhattacharya et al., 2003)
  - Stock market prediction (Tino et al., 2001)

Application of Recurrent Networks in Sequence Learning

- Sequence Classification
  - Classification of EEG signals (Forney & Anderson, 2011)
  - Visual pattern recognition: handwritten char. (Nishide et al., 2011)
  - Seismic signal classification (Park et al., 2011)
  - Pattern recognition in images (Abou-Nasr, 2010)
  - Emotion recognition from speech (Glüge et al., 2011)

- Sequence Prediction
  - Load forecasting in electric power systems (Barbounis et al., 2006)
  - Automatic speech processing (Varoglu & Hacioglu, 1999)
  - Sunspot series prediction (Park, 2011)
  - Network traffic prediction (Bhattacharya et al., 2003)
  - Stock market prediction (Tino et al., 2001)

Learning Long-Term Dependencies with Gradient Descent is Difficult (Bengio et al., 1994)

- *Vanishing Gradient Problem*
  - Recurrent networks (dynamic systems) have difficulties in learning a relationship between inputs separated over some time steps
  - Common learning algorithms based on computation of gradient information
  - Error signals tend to blow up or vanish
  - If blow up $\rightarrow$ network weights oscillate, if vanish $\rightarrow$ no learning
- Bengio et al. (1994) proved: condition leading to gradient decay is also a necessary condition for a dynamic system to robustly store information over longer periods of time

If the network configuration allows the storage of information over some time, then the problem of vanishing gradients appears.

Learning Long-Term Dependencies with Gradient Descent is Difficult (Bengio et al., 1994)

- *Vanishing Gradient Problem*
  - Recurrent networks (dynamic systems) have difficulties in learning a relationship between inputs separated over some time steps
  - Common learning algorithms based on computation of gradient information
  - Error signals tend to blow up or vanish
  - If blow up $\rightarrow$ network weights oscillate, if vanish $\rightarrow$ no learning
- Bengio et al. (1994) proved: condition leading to gradient decay is also a necessary condition for a dynamic system to robustly store information over longer periods of time

If the network configuration allows the storage of information over some time, then the problem of vanishing gradients appears.

Learning Long-Term Dependencies with Gradient Descent is Difficult (Bengio et al., 1994)

- *Vanishing Gradient Problem*
  - Recurrent networks (dynamic systems) have difficulties in learning a relationship between inputs separated over some time steps
  - Common learning algorithms based on computation of gradient information
  - Error signals tend to blow up or vanish
  - If blow up $\rightarrow$ network weights oscillate, if vanish $\rightarrow$ no learning
- Bengio et al. (1994) proved: condition leading to gradient decay is also a necessary condition for a dynamic system to robustly store information over longer periods of time

If the network configuration allows the storage of information over some time, then the problem of vanishing gradients appears.

Learning Long-Term Dependencies with Gradient Descent is Difficult (Bengio et al., 1994)

- *Vanishing Gradient Problem*
  - Recurrent networks (dynamic systems) have difficulties in learning a relationship between inputs separated over some time steps
  - Common learning algorithms based on computation of gradient information
  - Error signals tend to blow up or vanish
  - If blow up $\rightarrow$ network weights oscillate, if vanish $\rightarrow$ no learning
- Bengio et al. (1994) proved: condition leading to gradient decay is also a necessary condition for a dynamic system to robustly store information over longer periods of time

If the network configuration allows the storage of information over some time, then the problem of vanishing gradients appears.

- Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, TU Munich, 1991

Ways to Circumvent the Vanishing Gradient Problem

- Use learning algorithms that do not use gradient information
  - Simulated annealing
  - Cellular genetic algorithms
  - Expectation-maximization algorithm

- Variety of network architectures suggested, e.g.
  - Second-order recurrent neural network
  - Hierarchical recurrent neural network
  - Long short-term memory network (LSTM)
  - Echo state network

    ⋮

  - and Segmented-Memory Recurrent Neural Network (SMRNN)

Ways to Circumvent the Vanishing Gradient Problem

- Use learning algorithms that do not use gradient information
  - Simulated annealing
  - Cellular genetic algorithms
  - Expectation-maximization algorithm

- Variety of network architectures suggested, e.g.
  - Second-order recurrent neural network
  - Hierarchical recurrent neural network
  - Long short-term memory network (LSTM)
  - Echo state network

    ⋮
  - and Segmented-Memory Recurrent Neural Network (SMRNN)

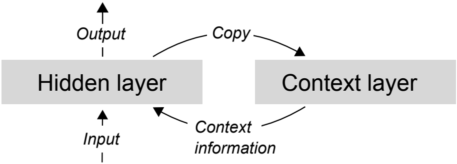
SMRNN Architecture (Chen & Chaudhari (2004))

Main idea:

- Inspired by process of memorisation of sequences observed in humans
- People fractionise long sequences into segments to ease memorisation
- For instance telephone numbers are broken into segments of digits
  $\rightarrow +493917214789$ becomes $+49\ 391\ 72\ 14\ 789$
- Behaviour is evident in studies of experimental psychology (Severin & Rigby, 1963; Wickelgren, 1967; Ryan, 1969; Frick,1989; Hitch et al., 1996)

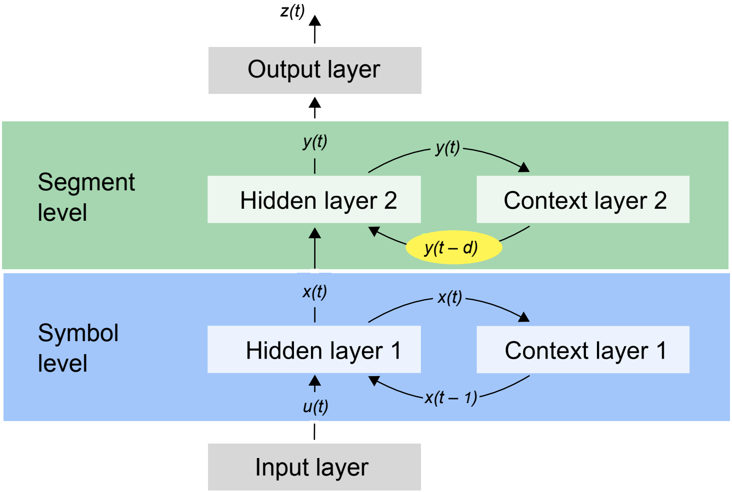
SMRNN Architecture (Chen & Chaudhari (2004))

- SMRNN is separated into:
  - *symbol level* (short-term information – single input)
  - *segment level* (long-term information – group of inputs)
- Each level realised by simple recurrent network (SRN)



- Two SRNs are arranged hierarchically

SMRNN Topology

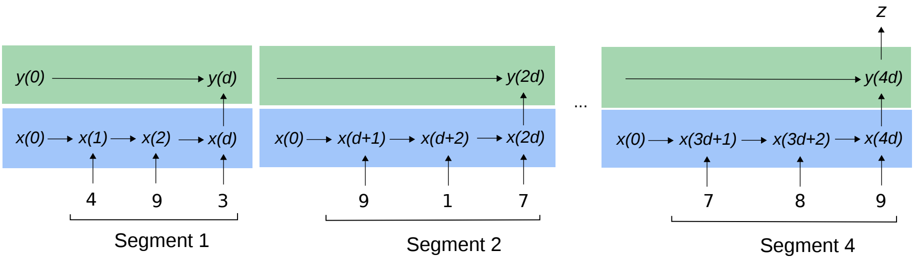


- difference between cascade of SRNs and SMRNN

  d – length of a segment may be fixed or variables

SMRNN Dynamics

- E.g. $+493917214789$ broken into segments of $d = 3$ digits



- Symbol level $x(t)$ updated after each input/digit
- Segment level $y(t)$ updated after $d = 3$ symbols
- Output $z(t)$ at the end of the sequence

Training of Recurrent Networks

Two popular training algorithms for recurrent networks

- Real-Time Recurrent Learning (RTRL) (Williams and Zipser, 1989)
  - Computes the exact error gradient at every time step
  - Computational complexity in order of $\mathcal{O}(n^4)$, n - number of network units in a fully connected network
- Backpropagation Through Time (BPTT) (Werbos, 1990)
  - "unfold" recurrent network in time by stacking copies of the network $\rightarrow$ feedforward network $\rightarrow$ backpropagation algorithm
  - Computational complexity in order of $\mathcal{O}(n^2)$

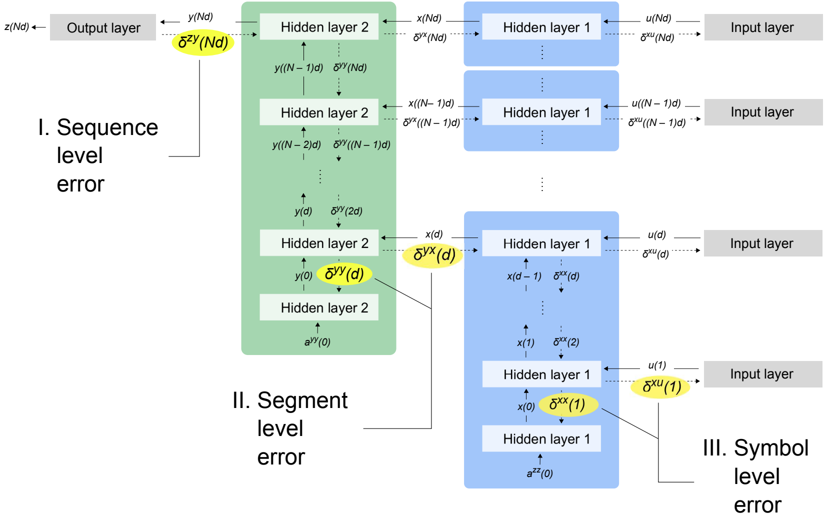
Extended Real-Time Recurrent Learning (eRTRL)

- Chen & Chaudhari (2004) adapted RTRL to SMRNNs $\rightarrow$ extended Real-Time Recurrent Learning (eRTRL)
- High computational complexity of RTRL also problem of eRTRL
- Makes it impractical for applications in big networks
- Time consuming training prevents parameter search for optimal number of hidden units, learning rate, etc.

Extended Backpropagation Through Time (eBPTT)

- Reduce computational complexity $\rightarrow$ adapt BPTT to SMRNNs
- Unfold SMRNN in time for one sequence
- Error at the end of a sequence propagated through unfolded network
- Segment level error computed only at *end of segment*
  $t = nd$ and $n = 0, \ldots, N$
- Symbol level error computed for *every time step/input*
  $t = 0, 1, 2, \ldots, dN$

Errorflow of eBPTT for a sequence of length *Nd*



Information Latching Problem

- Benchmark to test a system's ability to model long-term dependencies (Bengio et al., 1994)
- Task: classify sequences of length T, where *class* depends on first L items of the sequence
- Network needs to bridge $T - L$ time steps
- Example: phone number classification

$$+ \quad \underbrace{49}_{t-11,t-10} \quad \underbrace{3917214789}_{t-9,\dots,t} \Rightarrow \text{country?}$$

long-term dependency

- Country depends on inputs that lie 11 and 10 steps in the past

Information Latching Problem

- Sequences generated from an alphabet of 26 letters (a - z)
- Class label is provided at the end of sequence
- Task: classify strings of length T, where the *class* depends on a keyword (first L items)
- Example: sequence length $T = 22$, class-defining string $L = 10$

sequence	class C
p r e d e f i n e d r a n d o m s t r i n g	1
r a n d o m s t r i o m s t r i n g a b c d	0
h d g h r t z u s z j i t m o e r v y q d f	0
p r e d e f i n e d q u k w a r n g t o h d	1

Experimental Setup

- Predefined string $L = 50$, sequence length increased $T = 60, \dots, 130$
- 100 nets trained with eRTRL/eBPTT for every sequence length T
- Networks' configuration according to Chen & Chaudhari (2004)
  26 input, 10 symbol level, 10 segment level, 1 output,
  segment length $d = 15$,
  transfer function $f_{\mathrm{net}}(x) = 1/\left(1 + \exp(-x)\right)$
- Learning rate α and momentum η chosen after testing 100 networks on all combinations $\alpha \in \{0.1, 0.2, \dots, 0.9\}$ and $\eta \in \{0.1, 0.2, \dots, 0.9\}$ on the shortest sequence
- Training stopped when:
  - MSE $< 0.01 \rightarrow$ successful
  - or after 1000 epochs $\rightarrow$ unsuccessful

eBPTT vs. eRTRL on Information Latching

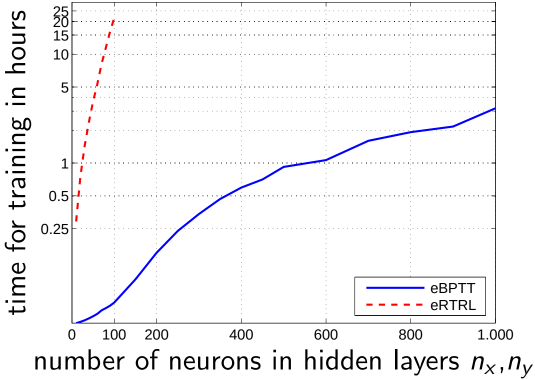
- #suc of 100 – number of successfully trained networks
- #eps – mean number of training epochs
- ACC – mean accuracy on test set

seq. length	eBPTT			eRTRL		
T	#suc	#eps	ACC	#suc	#eps	ACC
60	79	230.6	0.978	100	44.3	0.978
70	58	285.7	0.951	100	63.9	0.861
80	61	215.2	0.974	100	66.2	0.862
90	48	240.4	0.951	100	52.4	0.940
100	43	241.4	0.968	100	82.1	0.778
110	36	250.0	0.977	100	69.6	0.868
120	17	305.4	0.967	100	56.7	0.950
130	14	177.6	0.978	96	101.4	0.896
mean		243.3	0.968		67.1	0.892

Glüge et al., Extension of BPTT for Segmented-memory Recurrent Neural Networks, Proc. of Int. Joint Conf. on Comp. Intelligence (NCTA 2012)

Computation time

- Training for 100 epochs with 50 sequences of length $T = 60$
- Increase number of hidden units $10, \ldots, 1000$
- 100 hidden units: 3 minutes (eBPTT) vs. 21.65 hours (eRTRL)



eBPTT vs. eRTRL on Information Latching

- Decrease of successfully trained networks for eBPTT
- Nearly all networks were trained successfully with eRTRL

$\rightarrow$ **eRTRL was generally better able to cope with longer ranges of input-output dependencies.**

- Performance of trained networks (ACC) is higher for eBPTT
- Overall accuracy of 96.8% eBPTT compared to 89.2% eRTRL

$\rightarrow$ **Successful learning with eBPTT guaranteed better generalisation.**

- Computational complexity of eRTRL $\rightarrow$ impractical for large networks

eBPTT vs. eRTRL on Information Latching

- Decrease of successfully trained networks for eBPTT
- Nearly all networks were trained successfully with eRTRL

→ **eRTRL was generally better able to cope with longer ranges of input-output dependencies.**

- Performance of trained networks (ACC) is higher for eBPTT
- Overall accuracy of 96.8% eBPTT compared to 89.2% eRTRL

→ **Successful learning with eBPTT guaranteed better generalisation.**

- Computational complexity of eRTRL → impractical for large networks

eBPTT vs. eRTRL on Information Latching

- Decrease of successfully trained networks for eBPTT
- Nearly all networks were trained successfully with eRTRL

$\rightarrow$ **eRTRL was generally better able to cope with longer ranges of input-output dependencies.**

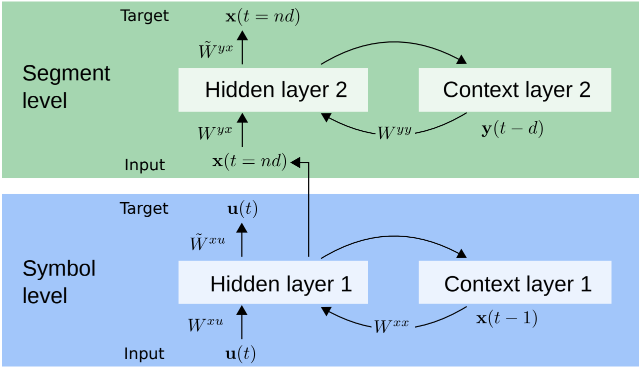
- Performance of trained networks (ACC) is higher for eBPTT
- Overall accuracy of 96.8% eBPTT compared to 89.2% eRTRL

$\rightarrow$ **Successful learning with eBPTT guaranteed better generalisation.**

- Computational complexity of eRTRL $\rightarrow$ impractical for large networks

Unsupervised Layer-local Pre-training

- SRNs on symbol/segment level separately trained as auto-encoder
- Symbol level SRN trained on the input data $\mathbf{u}(t)$
- Segment level SRN trained on symbol level output
  $\rightarrow$ for segment length d $\mathbf{x}(t = nd)$



eBPTT: Random Initialised and Pre-trained

- #suc of 100 – number of successfully trained networks
- #eps – mean number of training epochs
- ACC – mean accuracy on test set

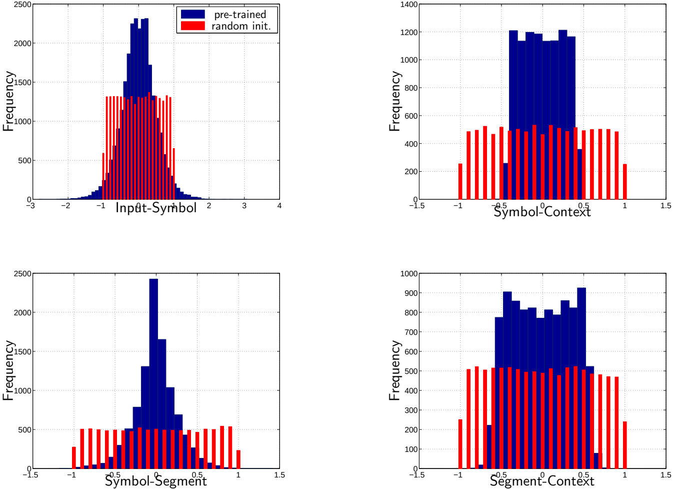
seq. length	randomly initialised			pre-trained		
T	#suc	#eps	ACC	#suc	#eps	ACC
60	80	122.6	0.966	91	69.8	0.968
70	83	80.3	0.962	96	41.9	0.971
80	65	123.3	0.968	95	31.3	0.979
90	41	180.3	0.978	77	29.4	0.977
100	37	147.1	0.971	82	40.3	0.979
110	26	204.2	0.980	75	55.6	0.981
120	16	239.6	0.954	49	32.4	0.987
130	6	194.8	0.987	52	39.2	0.977
mean	44.5	161.5	0.972	77.1	42.5	0.977

Glüge et al., Auto-Encoder Pre-Training of Segmented-Memory Recurrent Neural Networks, Proc. of the European Symposium on Art. NN, (ESANN 2013)

Conclusion: Layer-local Pre-training of SMRNNs

- Accuracy on test set (ACC) not influenced by pre-training
- Decrease of successfully trained networks with increasing sequence length T
- Pre-trained networks did not suffer from that behaviour as much as the randomly initialised
- $T = 130$, **52** pre-trained vs. **6** randomly initialised (out of 100) $\rightarrow$ pre-training improved eBPTT's ability to learn long-term dependencies

Weight Distribution: pre-trained vs. random initialised



- Pre-training effects *only* forward connections

Reset of context weights to zero

seq. length T	pre-trained #suc	pre-trained #eps	pre-trained ACC	pre-trained, Context $\rightarrow$ 0 #suc	pre-trained, Context $\rightarrow$ 0 #eps	pre-trained, Context $\rightarrow$ 0 ACC
60	89	53.7	0.964	97	47.8	0.964
70	98	43.8	0.975	96	33.0	0.972
80	90	37.3	0.981	95	60.8	0.973
90	85	26.8	0.984	83	77.1	0.977
100	88	28.6	0.984	77	46.0	0.975
110	73	24.5	0.982	80	52.4	0.984
120	56	34.1	0.991	61	78.5	0.972
130	59	32.5	0.990	57	$65,7$	0.989
Mittelwert	79.8	35.2	0.981	80.8	57.7	0.976

- Reset of context weight has no effect

$\rightarrow$ Pre-training does not support the learning of temporal relations between inputs

Glüge et al., Learning long-term dependencies in segmented-memory recurrent neural networks with backpropagation of error, Neurocomputing, Vol. 141 (2014)

Reset of context weights to the identity matrix

seq. length T	pre-trained #suc	pre-trained #eps	pre-trained ACC	pre-trained, context $\rightarrow \mathbb{1}$ #suc	pre-trained, context $\rightarrow \mathbb{1}$ #eps	pre-trained, context $\rightarrow \mathbb{1}$ ACC
60	89	53.7	0.964	98	60.5	0.977
70	98	43.8	0.975	99	30.4	0.969
80	90	37.3	0.981	98	26.9	0.985
90	85	26.8	0.984	98	29.2	0.975
100	88	28.6	0.984	96	27.2	0.994
110	73	24.5	0.982	88	26.0	0.995
120	56	34.1	0.991	74	45.3	0.988
130	59	32.5	0.990	74	47.6	0.994
Mittelwert	79.8	35.2	0.981	90.6	36.6	0.985

- Identity matrix supports error backpropagation

$\rightarrow$ Helps to solve the Information Latching Problem

Glüge et al., Learning long-term dependencies in segmented-memory recurrent neural networks with backpropagation of error,

Neurocomputing, Vol. 141 (2014)

Sequence Learning in RNNs

- Recurrent networks suffer the problem of vanishing gradients
- SMRNNs were proposed to circumvent the problem
  - eRTRL is computationally impractical for large networks
  - → eBPTT as an alternative: leads to better generalisation, but less able to catch long-term dependencies
- Pre-training of SMRNNs improves learning of long-term dependencies with eBPTT significantly

Emotion Recognition from Speech

- Identify emotional or physical state of a human from his/her voice
- Emotional state of a user helps to derive the semantics of a spoken sentence $\rightarrow$ enables the machine to respond in an appropriate manner (e.g. adapt dialogue strategy)
- Large range of classifiers was used for this task
  - Hidden Markov Models (HMMs) (Nwe et al., 2003; Song et al., 2008; Inoue et al., 2011)
  - Support Vector Machines (Pierre-Yves, 2003; Schuller et al., 2009)
  - Neural network: FFN (Nicholson et al., 1999; Petrushin, 2000), LSTM Networks (Wöllmer et al., 2008) and Echo State Networks (Scherer et al., 2008; Trentin et al., 2010)

Emotional Speech Database (EMO-DB)

Corpus:

- Ten predefined German sentences not emotionally biased by their meaning, e.g., "Der Lappen liegt auf dem Eisschrank."
- Sentences are spoken by ten (five male and five female) professional actors in each emotional way

EMO-DB utterances grouped by emotional class and separation into training/testing or training/validation/testing

Emotion	No. utterances	HMM	SMRNN
Anger	127	114/13	102/13/12
Boredom	79	71/8	63/8/8
Disgust	38	34/4	30/4/4
Fear	55	50/5	44/6/5
Joy	64	58/6	51/6/7
Neutral	78	70/8	62/8/8
Sadness	52	47/5	42/5/5

Feature extraction for HMM/SMRNN

- Speech data was processed using 25ms Hamming window
- with frame rate: 10ms HMM / 25ms SMRNN
- Each frame (25ms audio material) $\rightarrow$ 39 dimensional feature vector
  - 12 MFCCs + 0th cepstral coefficient (HMM/SMRNN)
  - first (Δ) and second derivatives ($\Delta\Delta$) (HMM)
- mean length of utterance in EMO-DB is 2.74s

$\rightarrow$ 10ms frame rate yields $274 \cdot 39 = 10686$ values per utterance

Classifier

- HMM
  - One model per class
  - Each model had 3 internal states (standard in speech processing)
  - Training and testing utilised the Hidden Markov Toolkit (Young et al., 2006)
- SMRNN
  - One network per Class
  - Each network differs in hidden layer units n_x, n_y, and segment length d (determined on validation set)

Emotion	n_x	n_y	d
Anger	28	8	17
Boredom	19	8	14
Disgust	22	14	8
Fear	17	17	7
Joy	19	29	2
Neutral	8	26	19
Sadness	13	13	11

Classifier

- HMM
  - One model per class
  - Each model had 3 internal states (standard in speech processing)
  - Training and testing utilised the Hidden Markov Toolkit (Young et al., 2006)
- SMRNN
  - One network per Class
  - Each network differs in hidden layer units n_x, n_y, and segment length d (determined on validation set)

Emotion	n_x	n_y	d
Anger	28	8	17
Boredom	19	8	14
Disgust	22	14	8
Fear	17	17	7
Joy	19	29	2
Neutral	8	26	19
Sadness	13	13	11

Results

Weighted and unweighted average (WA/UA) of class-wise accuracy in % for HMM and SMRNN classifiers

Emotion	training WA/UA	test WA/UA
SMRNN	91.08/91.62	71.02/73.47
HMM$\Delta\Delta$	79.70/81.76	73.75/77.55
HMMΔ	81.17/81.08	60.03/63.27
HMM	71.15/70.72	51.72/55.10

Results

Confusion matrix SMRNN classifier on test set with class-wise accuracy in % (Acc.)

Emotion	A	B	D	F	J	N	S
Anger	12	0	0	1	2	0	1
Boredom	0	5	0	0	0	3	0
Disgust	0	0	3	0	0	0	0
Fear	0	0	0	3	1	0	0
Joy	0	0	1	0	4	0	0
Neutral	0	1	0	1	0	5	0
Sadness	0	2	0	0	0	0	4
Acc.	100	62.5	75	60	57	62.5	80

Conclusion

- SMRNNs have the potential to solve complex sequence classification tasks as in automatic speech processing
- Networks are able to learn the dynamics of the input sequences $\rightarrow$ not necessary to provide the dynamic features of speech signal to learn the task
- SMRNNs performed slightly worse ($\approx$ 3% on test set) compared to HMMs
- Speech signal was sampled more frequently during feature extraction for HMMs (10ms for HMMs vs. 25ms for SMRNNs) $\rightarrow$ in total HMM$\Delta\Delta$ used 7.5 times more data than SMRNNs

Thank you for your attention!