

Good morning ladies and gentleman, welcome to the presentation of my master thesis, on which i worked together with MR.Lukas Mauch for the past half year. In the coming 30min i'll give a topic about "Deep neural network for speech emotion recognition".

As we know most of the current work of machine learning field has dedicated to computer vision and natural language processing and in NLP the linguistic information is recognized whereas the paralinguistic is discarded. But a more natural human-machine interaction requires also paralinguistic information where the personality of the speaker can also be recognized, such as age, gender and emotion.

To deal with emotion recognition we should firstly recognize the speech emotion data in the view of pattern recognition, where the emotion data is considered as

The state of the art in emotion recognition mostly based on GMM-HMM, where GMM tries to model the data distribution of speech data, however the work requires a lot of hand-crafted features and the number of component of GMM is generally restricted due to computation cost.

In order to model the speech emotion sufficiently, we exploit a technique called deep learning, which is a quite new field of machine learning and is currently widely discussed.

With deep architecture we can extract complex structure and building internal representations via unsupervised learning, those ideas have seen success in vision and audio processing

The content of this talk can be divided into following parts. Firstly the foundation of emotion recognition is shortly introduced which followed by second chapter to talk about CRBM in detail. And, in the third chapter i going to give a introduction to deep neural network regarding its basic structure and functions and how they are trained. Afterwards the Long short term memory for sequential modelling is discussed and then the result of this work is showed. Finally i am going to draw a short conclusion and give a outlook of the future research.

MFCC is one of the most commonly used features in speech emotion recognition.

The ...spectrum is calculated ... and transformed with the following equation to mel-scale in order to

The framework of emotion recognition is illustrated in the figure ...
Then those are the foundation of this work and in the next we are going to talk about CRBM

A RBM defines a basic structure showed with this figure. It has one visible layer of binary units denoted as vector x for receiving input data and one hidden layer of binary units denoted as vector h . for representations. each unit in visible layer is connected to each unit in hidden layer and the inter-layer connection is specified with the weight matrix W . b and c denotes the visible and hidden bias vector respectively.

Notice that the restriction of this model is that there is no intraconnection within each layer that is also the reason why this is called boltzmann machine.

To extract the emotion representations from the pre-processed MFCC features we build a temporal model named CRBM

In order to understand how this works we firstly take a look at the basic concept called RBM, which... P of a set of training data x , and probability distribution has some parameters to be learned during the training.

RBM is trained in ...

RBM is

There are several important definitions for understanding RBM.

The first concept is Energy FUNCTION called E subscript θ , θ denotes the parameter set W, b and c . The energy function defines the structure of the RBM, in other words the is related to how the visible and hidden layer connected with each other.

Then we have a joint distribution P of x, h, \dots , and Z is called partition function in order to normalize the right-hand side term so that it should be a sufficient probability distribution.

from the definition of the probability distribution we can see that if a higher probability is desired, the energy function should be small. if you are familiar with thermodynamic, you may have heard about Boltzmann distribution which tells us about the state of the system depending on its energy. With lower energy the system is by definition more stable and this idea is borrowed to build RBM.

Another concept Free Energy is defined and will be used in training RBM, we are going to cover this part later.

The inference of RBM is quite straightforward it requires only the mathematical calculation where the marginal distribution is calculated and using Bayes' theorem we can then get the conditional distribution. Then it allows us to calculate the probability of one particular taking value one given the corresponding input or hidden vector.

The RBM is a static model. However the emotion in speech varies over both short and long time period and consequently we need a temporal model to capture those short and long term variation. Therefore before we come to the training of the energy-based model, I am going to introduce an extension of RBM named CRBM. The matrix A and B are weight parameters of history visible units to current vis, and history visible units to current hidden units. Now the visible units are same as RBM the energy function defines how the units connected with each other, where \tilde{b} is defined as the original bias of visible layer plus the weight matrix A times history visible units, x subscript smaller than t , and the number of history input is specified by N , as shown in the figure intop. similarly case for the hidden bias \tilde{c} . now the parameter set consists ofand the free energy is also changed.

Now let's see how can the energy-based model be trained with MLE. Firstly I am going to introduce the KL-div, which defines the difference between two probability distributions by calculating the follow equation, here the integral is used when we have a continuous distribution and for the case of crbm where the distribution is discrete, the integral is substituted by the sum over x . As previous discussed, we want to use a model to approximate the true data distribution, here the $Q \dots P$ is and the brackets denotes the ... the first term on the right hand side is the distribution of the data, it can be treated as a constant value during the optimization so the objective remains only the second term

With the definition of free energy we can now rewrite the likelihood distribution its partial derivative with respect to the parameter set. if we average the derivative over the data, the objective function —(how become expectation over ...???) but the problem is that the calculation of the second term on the righthand side requires all configuration of the model which makes it impossible to be done directly, so we need to do sampling sufficiently to approximate the model distribution.

the technique used for sampling is Gibbs sampling where it performs a markov chain starting from the visible layer at time 0 and the hidden state at time 0 can be calculated with conditional probability, then the visible at time 1 is again calculated with conditional probability.. so this is a full step of Gibbs sampling.
The gibbs step repeats up to k full steps, where k is a pre-defined variable.

With $k=0$, the probability represents the distribution of input data which is independent of parameter set θ .
if Gibbs-steps is performed with big enough k steps e.g. up to infinite then the markov chain is guaranteed to converge to \dots : [click] so the model distribution can be approximated with the sampled distribution and the objective function can be rewritten as:

But in practise we cannot run the steps waiting until the chain converges and instead of optimizing the divergence between 0 step and infinite step, another concept is introduced by Hinton named Contrastive Divergence where the difference of KL between P_0 and P_{∞} and KL between P_1 and P_{∞} is minimized by calculating the following equation. It has already been known from many experiments that by running one full gibbs step we can already get good approximation of model distribution
From the literature by Hinton, we know that the third term of righthand side is irrelevant in optimization, so we can omit it during the training and simplify the CD.
Then we have the updating rule for the parameter set.

So in the next we are going to see the Deep Neural Network. The origin of artificial neural network trace back to the 60 and 70 in last century. To immitate how the human brain works, the scientists have used the artificial neurons and connections to build up the structure of neural network which is illustrated with the figure on the right. The structure shows a single hidden layer feedforward neural network. The data are fed forward from input layer on the bottom and through a hidden layer then to the output layer on the top.
the preactivation a of x of the hidden layer can be calculated by multiplying the input vector with weight matrix W superscript 1, adding the bias vector to the product.
mapping the preact a by a non-linear activation function f we can calculate the activation of the hidden layer.
similarly the activation of the output layer hat y is calculated based on the hidden layer. The output layer can be prediction or classification or fed as input of further network, depending on different activation function.
If the number of hidden units is more than 1, then we'll have a multi-layer neural network or DNN.

Training the neural network is generally done via supervised learning to minimize the empirical risk, which is shown in the first equation. It is the average of loss over a data sequence of size M and plus some regularization term $\lambda \gamma$ of θ . In the optimization the first-order technique gradient descent is most commonly used, where the gradient of each activation wrt to the parameter θ is calculated and backpropagated through the whole network. In practice the normal batch-gradient requires too much computation so in general the stochastic and mini ... are applied.

It has been a big topic in machine learning for quite a long time that how to train a deeper structure sufficiently and effectively, since the deeper the network is, the more useful features can be extracted and the network should be more powerful in classification tasks. But the training of multi-layer neural network via BP suffers from vanishing gradient problem.

Obviously the training time increases..... and most important is that the From literature it is recognized that the gradient vanishing is caused by...., so to cope with this problem a technique named is applied.

The deep network is trained in every step each layer... where \hat{x} is the reconstruction from auto-encoder after pre-training step the ...entire network is fine-tuned with ...

With this figure the pre-training step is illustrated in detail. At step 1 the training data is firstly mapped to the hidden layer to extract some features and then mapped again to output layer for reconstruction the input. After finishing minimize the reconstruction error for the first hidden layer the training is then fed to the pre-trained first hidden layer and from bottom up mapped to the second hidden layer and build the reconstruction of the first hidden layer. Now the optimization objective is the second layer, in this case the first hidden layer is treated as input and the recon. error is minimized wrt the first hidden layer.

This step repeats by feeding the training data directly to the pre-trained layer and minimize the corresponding reconstruction until all the hidden layer of the network is pretrained. Next is to do the supervised training, now BP won't suffer from the gradient vanishing problem since the parameter is good initialized with pre-training step.

Same as all other models the dnn should face with the overfitting problem. The dnn often have a large size first hidden layer and deep structure, so obviously due to the huge amount of parameters and lack of enough training data the training result may easily end up overfitting and poor generalization.

The overfitting problem can be solved by adding a regularization term to the loss function to penalize the weight with a hyper-parameter λ . In convex optimization adding a regularization term is equivalent to minimize the loss function with constrain on the p-norm of weight parameter.

The p-norm is defined as follows. In practice, using $p=1$ and $p=2$ can already prevent overfitting, and the corresponding regularization terms are named $L1$ and $L2$ regularization. The contour of $L1$ and $L2$ are shown separately in the figure, where the blue line are the contour of unregularized loss function, the origin of the circle indicates the minimum. the region surrounded by red edges are the constrain of $L2$ on the left and $L1$ on the right. the optimum value of weight is labeled as w^* . From the figure we can see that the $L1$ regularization can enable sparsity of the parameter because the

optimal value generally exist at the corner of the rotated square and consequently one of the parameter is zero.

So far we have talked about the CRBM and DNN, those belongs to the feedforward network now we are going to a new type of network which is developed from recurrent neural network— the LSTM.

The ordinary RNN suffers the same vanishing gradient problem during the BPTT, when the recurrent connection goes back to larger time steps in the past, so although it has great potential to model arbitrary non-linear dynamic system, but not much real application has seen since its existence.

Until 1997 xx and xx proposed a new variant of RNN called LSTM and solve the problem of training RNN by designing a special block of the RNN unit showed in this figure.

The block contains a memory cell c_t , whose activation is controlled by three gate units: input gate i forget gate f and the output gate o . The input gate controls when the newly arrived information from visible units is allowed to overwrite the current activation of memory cell, depending on two states of the input gate either "open" or "closed".

The activation of the memory cell is controlled by the forget gate : with gate being switched on, the information in memory cell remains unchanged with the fixed self-connection of weight 1, i.e. information is stored; while with gate being switched off, the information is discarded and the cell is reset to the initial state.

All the gate are trained to learn when it should be open or closed during the propagation.

With the forget gate The memory cell can keep the information unchanged and this is called constant error carousel With such the special memory block, LSTM is capable to preserve the long-time dependence by maintaining the gradients over time This is the central feature of LSTM.

With the figure below, lets see how this works. At time step 1, the input arrives at the cell, the input gate is on, then it can overwrite the current activation. and the the forget gate is on, so the information is stored by the memory cell, and the output gate is off so it wont be read by other other units. this was kept until step 6, the output gate on, the information is read by other units ,and then in time step 7 the forget and output is closed, old information is discarded.

So much for the deen network model, now we come to the experiment to see the performance of different models. For the classification task we used the EmoDB database, which is an online available database provided by TU Berlin. There are in total 7 emotions, and we use only four of them for test the performance of the model.

Schluss wort.