Deep Neural Network for Speech Emotion Recognition

—A Study of Deep Learning—

Zhuowei Han

Institut für Signalverarbeitung
und Systemtheorie

Universität Stuttgart

05.06.2014

Motivation

- Most current work focuses on speech processing based on linguistic information, e.g.: Skype Translator

- More natural human-machine interaction requires paralinguistic information such as age, gender, emotion.

- Speech Recognition / Speeker Identification / Emotion Recognition

Motivation

- Most current work focuses on speech processing based on linguistic information, e.g.: Skype Translator
- More natural human-machine interaction requires paralinguistic information such as age, gender, emotion.
- Speech Recognition / Speeker Identification / Emotion Recognition

Table of Contents

Table of Contents

Mel Frequency Cepstral Features

- Yoshua Bengio: A set of algorithms in machine learning that use a set of non-linear transformations to model high-level abstractions and hidden dependencies in data

Mel Frequency Cepstral Features

- Yoshua Bengio: A set of algorithms in machine learning that use a set of non-linear transformations to model high-level abstractions and hidden dependencies in data

A natural Deep Architecture

- Can learn high-level abstractions from unlabeled data
- Representationally efficient

Table of Contents

Structure of Deep Neural Networks

Computing net-activation

$$\begin{aligned}\underline{z}_k^{(l+1)} &= \mathbf{W}^{(l)}\underline{a}_k^{(l)} + \underline{b}^{(l)} \\ \underline{a}_k^{(l+1)} &= \underline{\Phi}\left(\underline{z}_k^{(l+1)}\right) \\ \underline{\hat{y}}_k &= \underline{a}_k^{(ol)}\end{aligned}$$

- Arbitrary non-linear mapping from $\underline{x}_k$ to $\underline{\hat{y}}_k$ possible
- Relation $N \Leftrightarrow$ Complexity
- Deep Architectures $(l\uparrow)$ more efficient than shallow ones $(l\downarrow, N_l\uparrow)$

Determining the parameters

Training objective

$$J(\mathbf{W}, \underline{b}) = \sum_{\forall k} \frac{1}{2} ||\underline{y}_k - \hat{\underline{y}}_k||^2 + \frac{\lambda}{2} \sum_{\forall l} ||\mathbf{W}^{(l)}||_F^2 \quad (1)$$

$$\mathbf{W}, \underline{b} = \arg\min_{\mathbf{W}, \underline{b}} J(\mathbf{W}, \underline{b}) \quad (2)$$

Numerical minimization

- Gradient calculation with Backpropagation
- Stochastic gradient descent
- **L**imited memory **B**royden-**F**letcher-**G**oldfarb-**S**hanno (L-BFGS)

Problems

- Optimization problem non-convex
  $\Rightarrow$ getting stuck in poor local minima
- Diffusion of gradients

Table of Contents

Unsupervised greedy layer-wise pre-training

- Train the Deep Neural Network layer by layer (Hinton, Bengio)
- Truncate network after first layer

- Reconstruction error

$$J_{AE} = \sum_{\forall k} \frac{1}{2} || \underline{a}_k^{(1)} - \underline{\hat{a}}_k^{(1)} ||^2$$

- Small hidden layer: Learned subspace similar to PCA for linear activation $\underline{\Phi}(\cdot)$

- Activation of the output layer
$\underline{\hat{a}}_k^{(1)} = \underline{\Phi}\left(\mathbf{W}^T \underline{\Phi}\left(\mathbf{W}\underline{x}_k + \underline{b}_{enc}\right) + \underline{b}_{rec}\right)$

Force non-trivial solution

- Reduce number of hidden neurons
- Regularization

$$J_{reg} = ||\mathbf{W}||_F^2 \tag{3}$$

- Sparsity constraint

$$\hat{\rho} = \frac{1}{m} \sum_{\forall k} [\underline{a}_k^{(2)}]_n \tag{4}$$

$$J_{sp} = \sum_{\forall n} \mathrm{KL}(\rho || \hat{\rho}_n) \tag{5}$$

$$\mathrm{KL}(\rho || \hat{\rho}_n) = \rho \log \frac{\rho}{\hat{\rho}_n} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_n} \tag{6}$$

- Overall cost

$$J = J_{AE} + \lambda J_{reg} + \beta J_{sp} \tag{7}$$

Unsupervised greedy layer-wise pre-training

- Propagate input to second layer

$$\underline{a}_k^{(2)} = \underline{\Phi}\left(\mathbf{W}^{(1)}\underline{a}_k^{(1)} + \underline{b}^{(1)}\right)$$

- Do pre-training of second layer
- ...

Unsupervised greedy layer-wise pre-training

- Add randomly initialized classification layer
- Perform dicriminative fine tuning, optimizing over weights and bias terms of each stage

Table of Contents

Experimental Setup

- Take 10 gray scale images
- Extract non-overlapping 8x8 patches
- Train Auto-Encoder for compression
- Setup of the Auto-Encoder
  - 1 hidden layer $[64, 25, 64]$
  - Training with 10.000 randomly selected patches
  - LBFGS for optimization

Data compression

Original

Reconstructed

Learned features

- Visualization
  - Plot row vectors of $\mathbf{W}^{(1)}$, because:
    $$\underline{z}_k^{(2)} = \mathbf{W}^{(1)}\underline{x}_k + \underline{b}^{(1)}$$
- The features are
  - Corner features
  - Edge features
  - Texture features

Experimental Setup

- Using MNIST data base
  - 60.000 binar training images
  - 10.000 binar test images
  - 28x28 pixels
- Setup of the dNN
  - 4 hidden layers $[784, 500, 200, 100, 10, 4]$
  - Sigmoid activation function in all layers
  - Tied-weights during layer-wise pre-training
  - Cost / gradient calculation with all 60.000 training sets
  - LBFGS for optimization

First stage features

Digit Recognition

Last stage features

$$\left[\underline{a}^{(ol)}\right]_n$$

Result

- Clustering into 16 groups
- Learned representations are prototypes of handwritten digits
- Recognition rate after discarding the last layer and performing discriminative fine tuning 98.2%

Experimental Setup

- Using MNIST data base
- Adding random distortion which flips values at arbitrary positions $\underline{\tilde{x}}_k = \underline{x}_k + \underline{w}$
- Setup of the Auto-Encoder
  - 1 hidden layer $[784, 196, 784]$
  - Sigmoid activation function in all layers
  - Tied-weights
  - Cost / gradient calculation with all 60.000 training sets
  - LBFGS for optimization

Results

Quadratic error:

$$
\begin{aligned}
e_1 &= \frac{1}{NL}\sum_{k=1}^{N} ||\underline{x}_k - \underline{\tilde{x}}_k||^2 = 0.0873 \\
e_2 &= \frac{1}{NL}\sum_{k=1}^{N} ||\underline{x}_k - \underline{\hat{y}}_k||^2 = 0.0158
\end{aligned}
$$

Results

Quadratic error:

$$
\begin{aligned}
e_1 &= \frac{1}{NL}\sum_{k=1}^{N} ||\underline{x}_k - \tilde{\underline{x}}_k||^2 = 0.2038 \\
e_2 &= \frac{1}{NL}\sum_{k=1}^{N} ||\underline{x}_k - \hat{\underline{y}}_k||^2 = 0.0239
\end{aligned}
$$

Auto-Encoder for image reconstruction

Why this works (Vincent et al. 2010)

- Auto-Encoder captures structure of input distribution
- Learns to map from low-probability regions to lower-dimensional high-probability regions

Table of Contents

Auto-Encoder for image reconstruction

- Deep Architectures can bridge the gap between feature engineering and classification (representation learning)
- Deep Architectures can learn hierarchical abstractions from high-dimensional raw data and therefore enable non-local learning
- Greedy layer-wise pre-training results in an initialization of the network near a good local minima of the cost function
- Only unlabeled data is used during pre-training
- Stacked Auto-Encoders can be used for reconstruction of noisy data (Maybe even for reconstruction of MR-Images??)