Deep Neural Networks

DT2119 Speech and Speaker Recognition

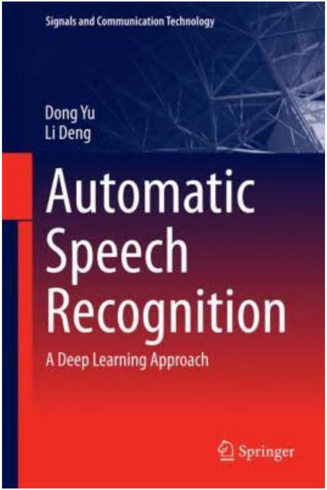
Giampiero Salvi

KTH/CSC/TMH giampi@kth.se

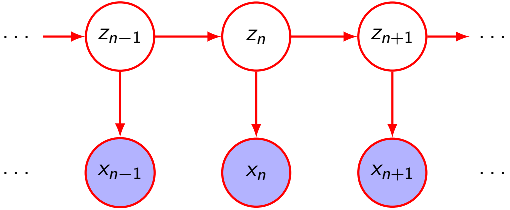
VT 2017

Literature

D. Yu and L. Deng. *Automatic Speech Recognition, a Deep Learning Approach*. Springer, 2015
Available in PDF through KTH Library


Signals and Communication Technology
Dong Yu
Li Deng
Automatic Speech Recognition
A Deep Learning Approach
Springer

State-to-Output Probability Model



is responsible for the discriminative power of the whole model

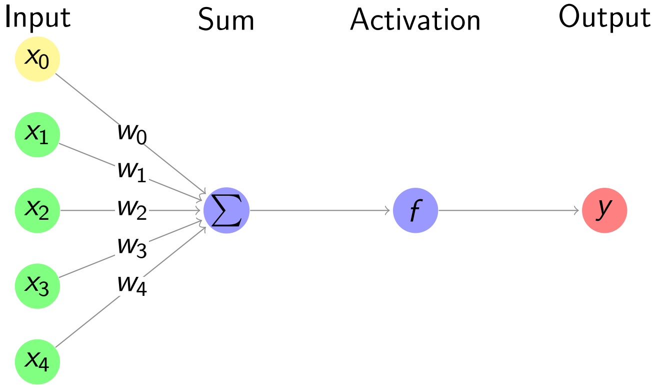
- GMMs used because easy to train and adapt
- discriminative training can improve results

Alternatives:

- artificial neural networks (ANNs)
- deep neural networks (DNNs)
- support vector machines (SVMs) not used for ASR

Perceptron

Known since the 1950's [17]

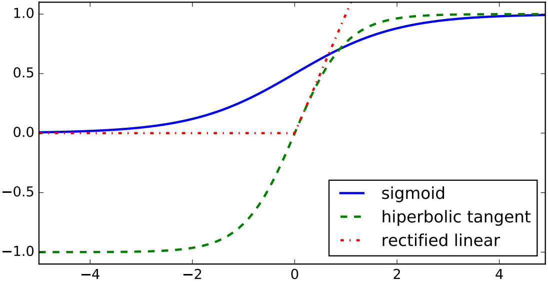


[17] F. Rosenblatt. *The perceptron: A perceiving and recognizing automaton*. Tech. rep. 85-460-1. Cornell Aeronautical Laboratory, 1957

Perceptron: Activation function

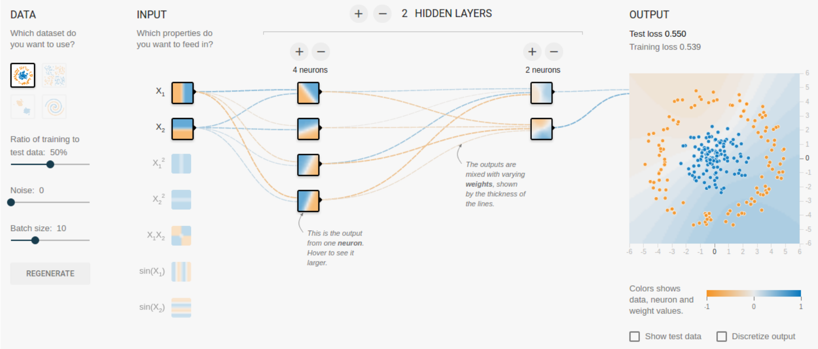
$$y = f\left(b + \sum_i w_i x_i\right)$$

$$\begin{aligned} f(z) &= \frac{1}{1+e^{-z}} && \text{sigmoid} \\ f(z) &= \frac{e^z - e^{-z}}{e^z + e^{-z}} && \text{hyperbolic tangent} \\ f(z) &= \max(0, z) && \text{rectified linear unit} \end{aligned}$$



Equivalent to logistic regression ($b = w_0 x_0$ bias)

Preceptron: Illustration



DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

INPUT

Which properties do you want to feed in?

2 HIDDEN LAYERS

4 neurons

2 neurons

This is the output from one ***neuron****. Hover to see it larger.*

The outputs are mixed with varying ***weights****, shown by the thickness of the lines.*

OUTPUT

Test loss 0.550

Training loss 0.539

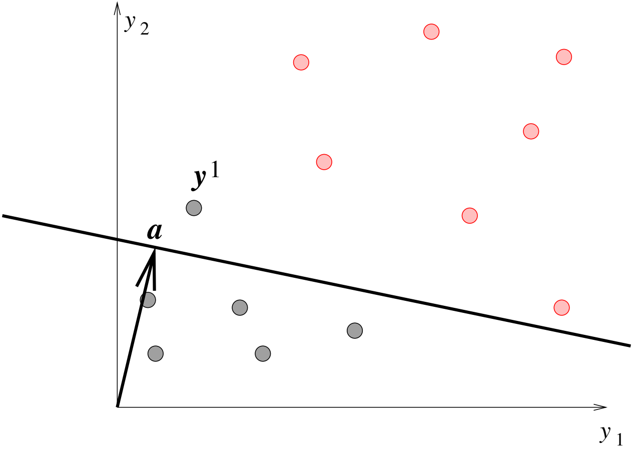
Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

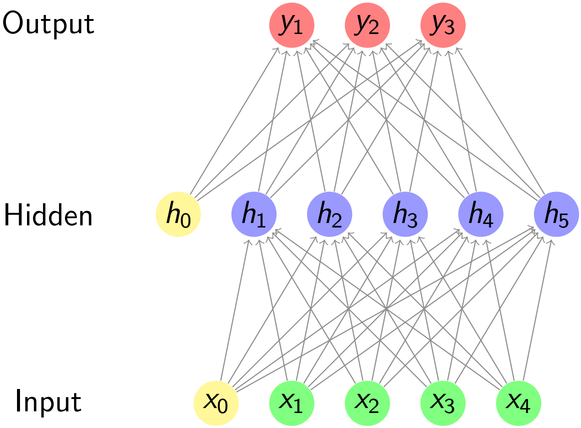
http://playground.tensorflow.org/

Perceptron: Linear Classification

Learning adjust weights to correct errors



Multi-layer Perceptron [16]



[16] F. Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms.* Tech. rep. DTIC Document, 1961

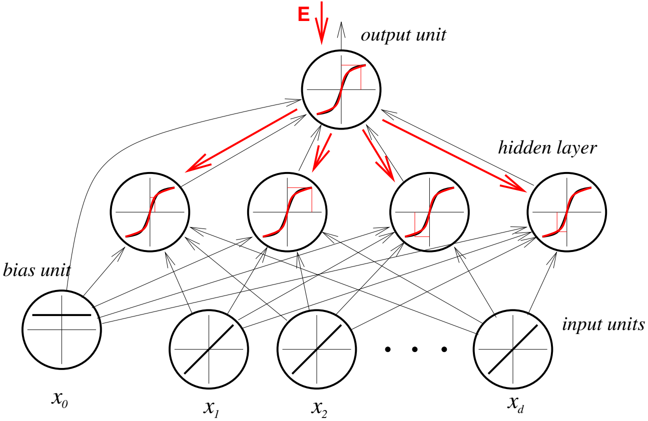
Universal Approximation Theorem

- First proposed by Gybenko [7]
- one single hidden layer and finite but appropriate number of neurons
- can approximate any function in $\mathbb{R}^N$ with mild constraints

[7] G. Gybenko. "Approximation by superposition of sigmoidal functions". In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314

Multi-layer Perceptron: Training

Backpropagation algorithm [11] [3] [18]



[11] H. J. Kelley. "Gradient Theory of Optimal Flight Paths". In: *ARS Journal* 30.10 (1960), pp. 947–954

[3] A. E. Bryson. "A gradient method for optimizing multi-stage allocation processes". In: *Proc. of the Harvard Univ. Symposium on digital computers and their applications*. 1961

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*. Tech. rep. DTIC Document, 1985

Learning Criteria

Ideally minimise Expected Loss:

$$J_{\mathrm{EL}} = \mathbb{E}\big[J(W, B, o, y)\big] = \int_o J(W, B, o, y)p(o)do$$

where o = features, y = labels

but we do not know $p(o)$

Use empirical learning criteria instead:

- Mean Square Error (MSE)
- Cross Entropy (CE)

Mean Square Error Criterion

$$J_{\text{MSE}} = \frac{1}{M}\sum_{m=1}^{M} J_{\text{MSE}}(W, B, o^m, y^m)$$

$$\begin{aligned} J_{\text{MSE}}(W, B, o^m, y^m) &= \frac{1}{2}\left\| v^L - y \right\|^2 \\ &= \frac{1}{2}\left(v^L - y \right)^T \left(v^L - y \right) \end{aligned}$$

Cross Entropy Criterion

$$J_{\mathrm{CE}} = \frac{1}{M} \sum_{m=1}^{M} J_{\mathrm{CE}}(W, B, o^m, y^m)$$

$$J_{\mathrm{CE}}(W, B, o^m, y^m) = -\sum_{i=1}^{C} y_i \log v_i^L$$

Equivalent to minimising Kullback-Leibler divergence (KLD)

Update rules

$$\begin{aligned} W^l_{t+1} &\leftarrow W^l_t - \epsilon \Delta W^l_t \\ b^l_{t+1} &\leftarrow b^l_t - \epsilon \Delta b^l_t \end{aligned}$$

To compute ΔW^l_t and Δb^l_t we need the gradient of the criterion function.
Key trick: chain rule of gradients $f(g(x))$:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$$

Backpropagation: Properties

- weights only depend on neighbouring variables
- algorithm finds local optimum
- sensitive to initialisation

Practical Issues

- initialisation: random (symmetry breaking), linear range of activation function
- regularisation (weight decay, dropout)
- batch size selection
- sample randomisation
- momentum
- learning rate and stopping criterion

Output Layer

Different from all other layers (adapted to the task)

Regression tasks: Linear layer

$$v^L = z^L = W^L v^{L-1} + b^L$$

Classification tasks: Softmax layer

$$v_i^L = \text{softmax}_i(z^L) = \frac{e^{z_i^L}}{\sum_{j=1}^{C} e^{z_j^L}}$$

Softmax: Probabilistic Interpretation

1. $v_i^L \in [0, 1] \ \ \forall i$
2. $\sum_{j=1}^{C} v_i^L = 1$

Output activations are posterior probabilities of the classes given the observations

$$v_i^L = P(i|o)$$

In speech: $P(\text{state}|\text{sounds})$

Hybrid HMM+Multi Layer Perceptron

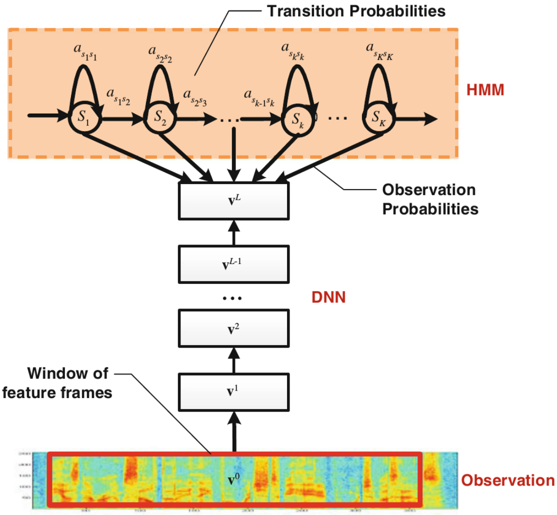


Figure from Yu and Deng

Combining probabilities [1]

- HMMs use likelihoods $P(\text{sound}|\text{state})$
- MLPs and DNNs estimate posteriors $P(\text{state}|\text{sound})$

We can combine with Bayes:

$$P(\text{sound}|\text{state}) = \frac{P(\text{state}|\text{sound})P(\text{sound})}{P(\text{state})}$$

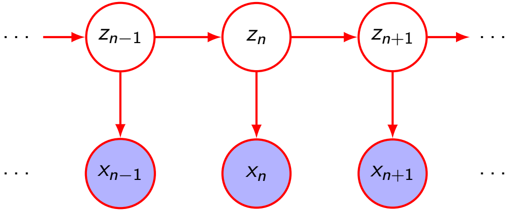
- $P(\text{state})$ can be estimated from the training set
- $P(\text{sound})$ is constant and can be ignored

Use scaled likelihoods:

$$\bar{P}(\text{sound}|\text{state}) = \frac{P(\text{state}|\text{sound})}{P(\text{state})}$$

[1] H. Bourland and C. J. Wellekens. "Links Between Markov Models and Multilayer Perceptrons". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 12.12 (1990)

State-to-Output Probability Model



Use ANNs for $P(x_n|z_n)$

Time-Delayed NNs [23]

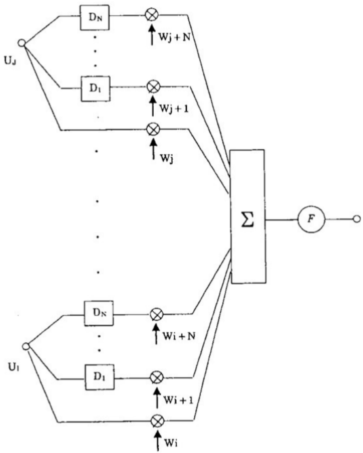
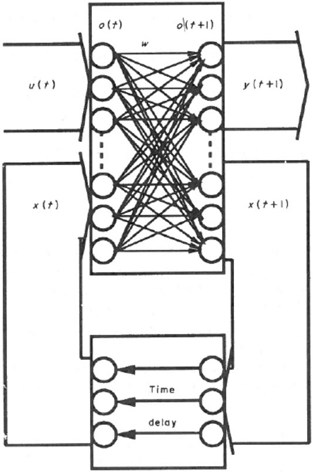


Fig. 1. A Time-Delay Neural Network (TDNN) unit.

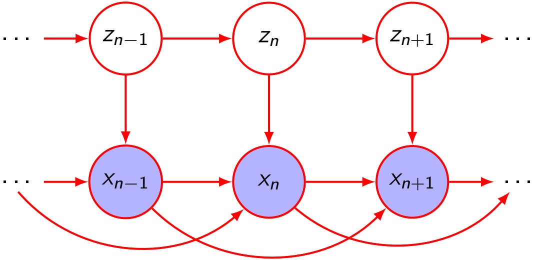
[23] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. "Phoneme Recognition Using Time-Delay Neural Networks". In: *IEEE Trans. Acoust., Speech, Signal Process.* 37.3 (1989)

Recurrent ANNs [15]



[15] T. Robinson and F. Fallside. “A recurrent error propagation network speech recognition system”. In: *Computer Speech and Language* 5.3 (1991), pp. 259–274

HMM + RNN Dependencies

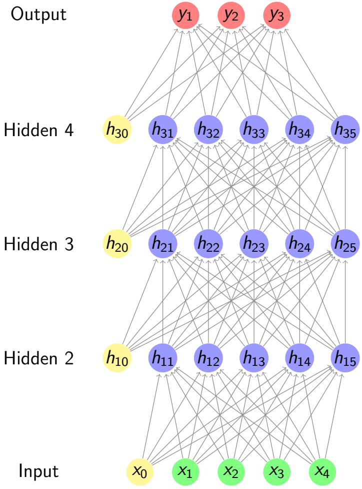


How do the two models interact? [20]

[20] G. Salvi. “Dynamic Behaviour of Connectionist Speech Recognition with Strong Latency Constraints”. In: *Speech Communication* 48.7 (July 2006), pp. 802–818

Deep Neural Network

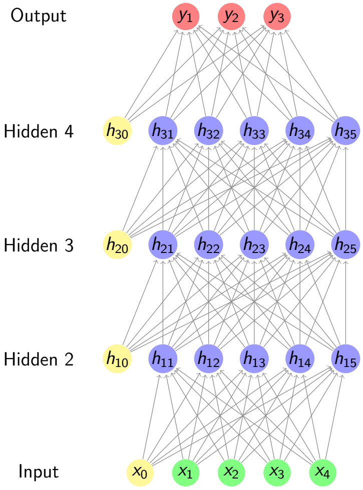
First appearance in 1965 [10]



[10] A. G. Ivakhnenko and V. G. Lapa. *Cybernetic Predicting Devices*. Purdue University School of Electrical Engineering, 1965

DNN: Motivation

- depth $\sim$ abstraction
- good initialisation (see later)
- reuse of features through the model
- fast computers, large datasets



DNN and MLPs

- no conceptual difference from MLPs
- limited use in the past due to limits in backpropagation
  - local minima
  - vanishing gradients
- brought to spotlight through pre-training [5]

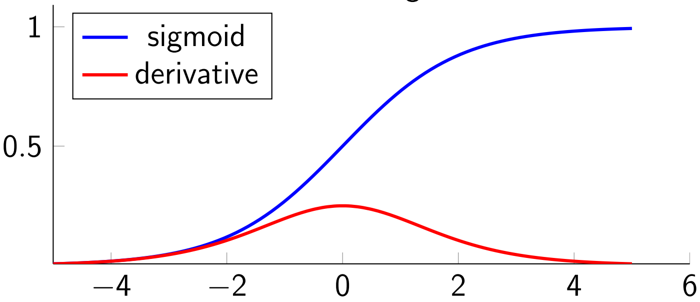
(later Backpropagation has been proven to be sufficient)

[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Mansagol, and P. Vincent. "Why Does Unsupervised Pre-training Help Deep Learning?" In: *Journal of Machine Learning Research* 11 (2010), pp. 625–660

Analysis of vanishing gradients [6] [12]

sigmoid and tanh function partly responsible for vanishing gradients

derivative of sigmoid



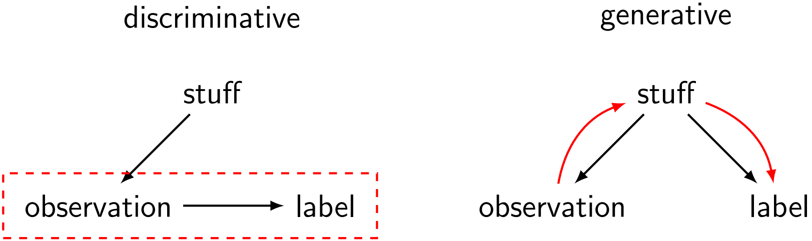
[6] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proc. AISTATS*. 2010

[12] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller. "Efficient backprop". In: *Neural networks, tricks of the trade*. 1998

Pre-Training in Deep Learning

- attempt to overcome limits of backpropagation
- pioneered by Geoffry Hinton (Univ. Toronto)
- unifies properties of generative and discriminative models

Pre-Training: Idea

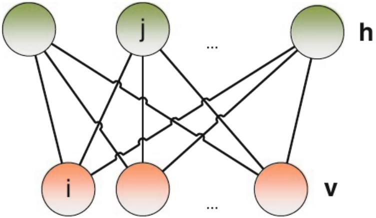


Deep Learning: Idea #2

1. initialise DNN with Restricted Boltzmann Machines (RBM) that can be trained unsupervised
2. use fast learning procedure (Hinton)
3. use ridiculous amounts of unlabelled (cheap) data to train a ridiculous number of parameters in an unsupervised fashion
4. at the end, use small amounts of labelled (expensive) data and backpropagation to learn the labels

Restricted Boltzmann Machines (RBMs)

First called Harmonium [21]

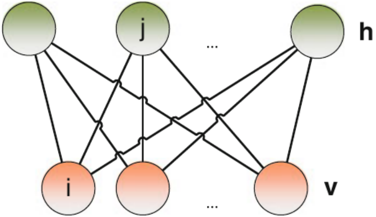


- binary nodes: Bernoulli distribution
- continuous nodes: Gaussian-Bernoulli

Figure from Yu and Deng

[21] P. Smolensky. "Information processing in dynamical systems: Foundations of harmony theory". In: Department of Computer Science, University of Colorado, Boulder, 1986. Chap. 6

Restricted Boltzmann Machines (RBMs)



Energy (Bernoulli):

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v}$$

Energy (Gaussian-Bernoulli):

$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2}(\mathbf{v} - \mathbf{a})^T(\mathbf{v} - \mathbf{a}) - \mathbf{b}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v}$$

RBM: Probabilistic Interpretation

$$P(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}$$

Posteriors (conditional independence):

$$P(\mathbf{h}|\mathbf{v}) = \cdots = \prod_i P(h_i|\mathbf{v})$$

and

$$P(\mathbf{v}|\mathbf{h}) = \cdots = \prod_i P(v_i|\mathbf{h})$$

Binary Units: Cond Prob

Posterior equals sigmoid function!!

$$
\begin{aligned}
P(h_i = 1|\mathbf{v}) &= \frac{e^{(b_i 1 + 1\mathbf{W}_{i,*}\mathbf{v})}}{e^{(b_i 1 + 1\mathbf{W}_{i,*}\mathbf{v})} + e^{(b_i 0 + 0\mathbf{W}_{i,*}\mathbf{v})}} \\
&= \frac{e^{(b_i 1 + 1\mathbf{W}_{i,*}\mathbf{v})}}{e^{(b_i 1 + 1\mathbf{W}_{i,*}\mathbf{v})} + 1} \\
&= \frac{1}{1 + e^{-(b_i 1 + 1\mathbf{W}_{i,*}\mathbf{v})}} \\
&= \sigma(b_i 1 + 1\mathbf{W}_{i,*}\mathbf{v})
\end{aligned}
$$

Same as Multi Layer Perceptron (viable for initialisation!)

Gaussian Units: Cond Prob

$$P(\mathbf{v}|\mathbf{h}) \;=\; \mathcal{N}(\mathbf{v}; \mu, \boldsymbol{\Sigma})$$

with

$$\begin{aligned} \mu &= \mathbf{W}^T\mathbf{h} + \mathbf{a} \\ \boldsymbol{\Sigma} &= \mathbf{I} \end{aligned}$$

RBM Training

Stochastic Gradient Descend (minimise the negative log likelihood)

$$J_{\mathrm{NLL}}(\mathbf{W}, \mathbf{a}, \mathbf{b}, \mathbf{v}) = -\log P(\mathbf{v}) = F(\mathbf{v}) + \log \sum_{\mathbf{v}} e^{-F(\mathbf{v})}$$

where

$$F(\mathbf{v}) = -\log \left(\sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \right)$$

is the free energy of the system.

BUT: the gradient can not be computed exactly

RBM Gradient

$$\frac{\partial J_{\text{NLL}}(\mathbf{W}, \mathbf{a}, \mathbf{b}, \mathbf{v})}{\partial \theta} = \frac{\partial F(\mathbf{v})}{\partial \theta} - \sum_{\tilde{\mathbf{v}}} p(\tilde{\mathbf{v}}) \frac{\partial F(\tilde{\mathbf{v}})}{\partial \theta}$$

- first term increases prob of training data
- second term decreases prob density defined by the model

RBM Stochastic Gradient

The general form is:

$$\begin{aligned}\nabla_\theta J_{\mathrm{NLL}}(\mathbf{W},\mathbf{a},\mathbf{b},\mathbf{v}) &= -\left[\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}\right\rangle_{\mathrm{data}} \right. \\ &\qquad \left. - \left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}\right\rangle_{\mathrm{model}}\right]\end{aligned}$$

Example: visible layer

$$\begin{aligned}\nabla_{w_{ij}} J_{\mathrm{NLL}}(\mathbf{W},\mathbf{a},\mathbf{b},\mathbf{v}) &= -\left[\langle v_i h_j\rangle_{\mathrm{data}}\right. \\ &\qquad \left. - \langle v_i h_j\rangle_{\mathrm{model}}\right]\end{aligned}$$

Gibbs Sampling

$\langle v_i h_j \rangle_{\text{model}}$ computed with sampling

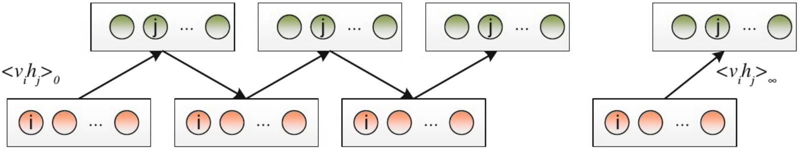
Sample joint distribution of N variables, one at a time:

$$P(X_i | X_{-i})$$

where X_{-i} are all the other variables

BUT: it takes exponential time to compute exactly

Contrastive Divergence



Two tricks:

1. initialise the chain with a training sample
2. do not wait for convergence

It turns out it is enough to go up and down once.

Deep Belief Networks

- We would like to stack several RBMs on top of each other
- The resulting model is called Deep Belief Network (DBN)
- Motivation: initialise Deep Neural Networks
- Problem: how to train them?

Deep Belief Networks: Training

Yee-Whye Teh (one of Hinton's students) observed that DBNs can be trained greedily

for each layer:

1. train a RBM unsupervised
2. excite the network with training data to produce outputs
3. use the outputs to train next RBM

RBMs and Deep Belief Networks

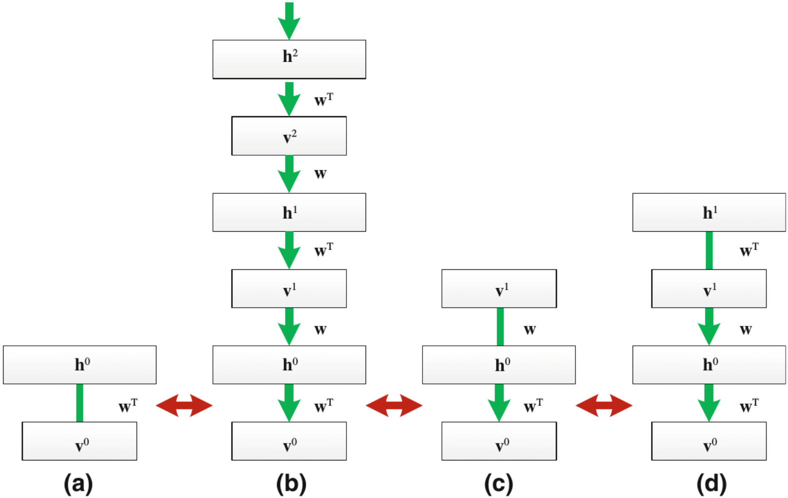


Figure from Yu and Deng

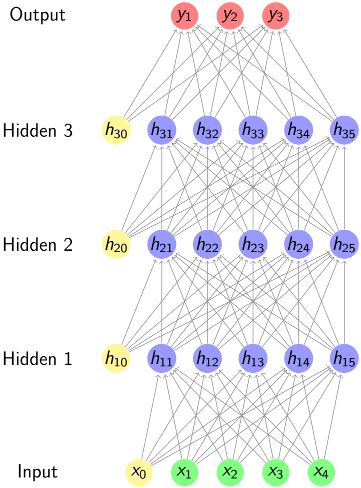
Training Deep Neural Networks

1. Train a Deep Belief Network unsupervised with Contrastive Divergence
2. use the DBN weights as initialisation of a Deep Neural Network
3. add an output layer to the DNN
4. retrain in a supervised way the whole DNN with backpropagation

Advantage:

- Supervised training starts close to a good optimum

Final Step: Supervised Training



The importance of pre-training

- Backpropagation alone not powerful enough [5]
- Use good initialisation and momentum [22]

Why?

- one reason: vanishing gradients [9]
- dependent on the activation function

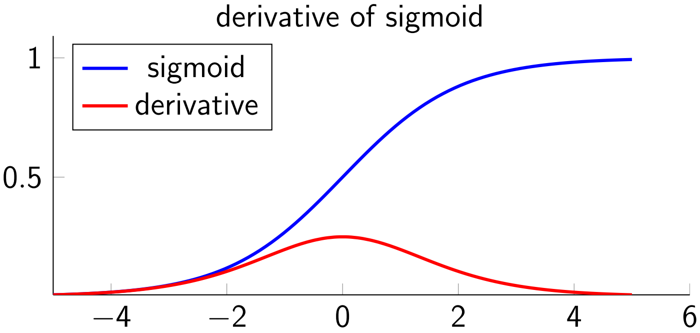
[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Mansagol, and P. Vincent. "Why Does Unsupervised Pre-training Help Deep Learning?" In: *Journal of Machine Learning Research* 11 (2010), pp. 625–660

[22] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. "On the importance of initialization and momentum in deep learning". In: *Proc. of ICML*. 2013

[9] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies". In: *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by S. C. Kremer and J. F. Kolen. IEEE Press

Analysis of vanishing gradients [6] [12]

sigmoid and tanh function partly responsible for vanishing gradients



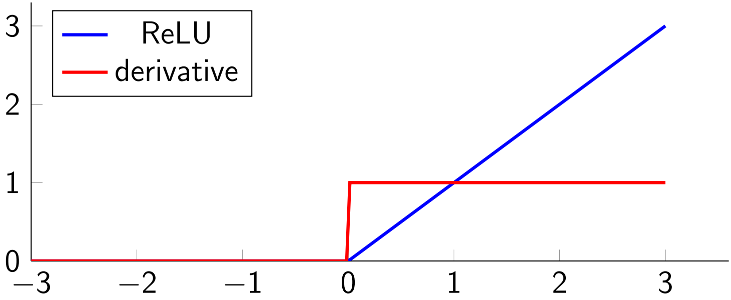
[6] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proc. AISTATS*. 2010

[12] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller. "Efficient backprop". In: *Neural networks, tricks of the trade*. 1998

Alternative Solution

Use better activation function [25]

Rectified Linear Unit (ReLU)



[25] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton. "On Rectified Linear Units for Speech Processing". In: *Proc. of IEEE ICASSP*. 2013

Applications to Speech

First modern:

- Phone Recognition (TIMIT) [4]
- 1 to 9 hidden layers
- 512, 1024, 1536, 2048 units/layer
- input: MFCC

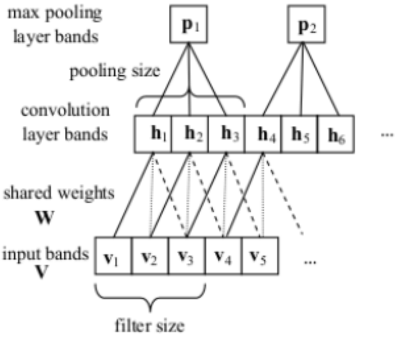
[4] G. E. Dahl, M. Ranzato, A.-r. Mohamed, and G. Hinton. "Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine". In: *NIPS*. 2010

Later extended to Large Vocabulary [8]

- University of Toronto, Microsoft, Google, IBM
- TIMIT, phone recognition
- Bing voice search
- Switchboard large vocabulary
- Google voice input
- YouTube speech recognition task
- English broadcast news
- both MFCC and Filterbank features

[8] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. "Deep Neural Networks for Acoustic Modeling in Speech Recognition". In: *IEEE Signal Processing Magazine* (2012)

Deep Convolutional Nets LVCSR



- IBM Watson Research Center, University of Toronto [19]
- English Broadcast News task

[19] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran. "Deep Convolutional Neural Networks for LVCSR". . In: *Proc. of IEEE ICASSP*. 2013

Deep Convolutional Nets LVCSR

work best with locally correlated features!

- no MFCCs, no LDA
- Mel Filterbank Features!!
- Vocal tract normalisation helps
- using deltas and delta-deltas helps

Differences/Similarities with Image ConvNets

Similarities

- several convolutional layers + several fully connected

Differences (low vs high frequencies)

- weight sharing only for neighbouring frequencies
- increase the number of hidden units

Typical Training Procedure

1. train a full context dependent GMM-HMM system
2. cluster CD HMM states into senones (order of 1000)
3. use senones to define output of DNN
4. run forced alignment with GMM-HMMs
5. train DNN with forced aligned transcriptions

Typical Features

- MFCCs + Deltas for GMM-HMMs
- Window of MFCCs + LDA for GMM-HMMs
- Filterbanks for DNNs

Features adaptation techniques (Later lecture):

- Vocal Tract Length Normalisation (VTLN)
- Feature Maximum Likelihood Linear Regression (fMLLR)

ANNs in ASR: Advantages

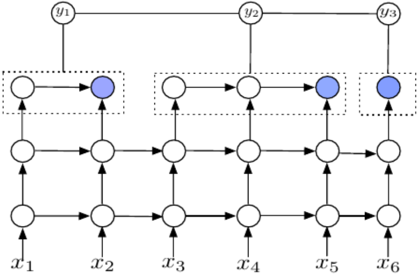
- discriminative in nature
- powerful time model:
- Time-Delayed Neural Networks (TDNNs)
- Recurrent Neural Networks (RNNs)

ANNs in ASR: Disadvantages

- training requires state level annotations (no EM available)
- usually annotations obtained with forced alignment (Viterbi training)
- not easy to adapt
- we still need GMM-HMMs for the training
- the input are still highly engineered features

. . . but . . .

Connectionist Temporal Classification

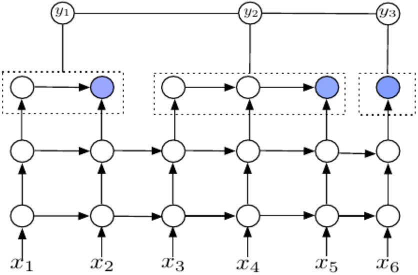


- End-to-end [13] (TIMIT), [14] (LibriSpeech)
  - no features
  - no segmentation (phone annotations)
  - no lexical model
- map speech samples to characters

[13] L. Lu, L. Kong, C. Dyer, N. A. Smith, and S. Renals. "Segmental Recurrent Neural Networks for End-to-end Speech Recognition". In: *arXiv:1603.00223* (2016)

[14] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur. "Purely sequence-trained neural networks for ASR based on lattice-free MMI". . In: 2016

Connectionist Temporal Classification



Advantages

- only orthographic transcriptions required
- no (expensive) lexical models
- never again an out-of-vocabulary word

Disadvantages

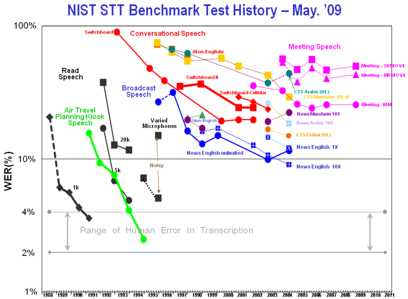
- still need to map (noisy) spellings to words

State-of-the-art

wer_are_we (pun with Word Error Rate)
`https://github.com/syhw/wer_are_we`

Database	Task	Style
Wall Street Journal (WSJ)	Large Vocabulary	Prompts
Switchboard (telephone)	Large Vocabulary	Spontaneous
TIMIT	Phoneme rec	Prompts

Risks of state-of-the-art in research



- overused databases
- meta optimisation on test sets
- focus on tiny improvements rather than new ideas
- in ANN terms: getting stuck in local minimum [2]

[2] H. Bourlard, H. Hermansky, and N. Morgan. "Towards increasing speech recognition error rates". In: *Speech Communication* 18 (1996)