Deep neural networks

Advanced data-mining

Yongdai Kim

Department of Statistics, Seoul National University, South Korea

Weakness of shallow structured model

- Shallow networks are very effective in solving many simple and well-constrained problems, but they may be deficient in capability of representing functions for solving complicated problems in real-world applications.
- This is because shallow networks may require an exponential number of computational units in dealing with some problems, whereas deep networks just need much fewer.(O.Delalleau and Y.Bengio, 2011)

Representational ability of deep structured model

Sum-product network

- Let's analyze the representational ability of a sum-product network.(O.Delalleau and Y.Bengio (2011)
- Suppose the input variables is $\mathbf{x} = (x_1, ..., x_n)$ and $n = 4^i$, where i is a positive integer.
- The units of odd layers and even layers can be constructed as follows:

$$\begin{cases} l_j^{(2k+1)} = l_{2j-1}^{(2k)} \cdot l_{2j}^{(2k)} & \text{for } 0 \leq k \leq i-1 \text{ and } 1 \leq j \leq 2^{2(i-k)-1} \\ l_j^{(2k)} = \lambda_{jk} l_{2j-1}^{(2k-1)} + \mu_{jk} l_{2j}^{(2k-1)} & \text{for } 1 \leq k \leq i \text{ and } 1 \leq j \leq 2^{2(i-k)} \end{cases}$$

Representational ability of deep structured model

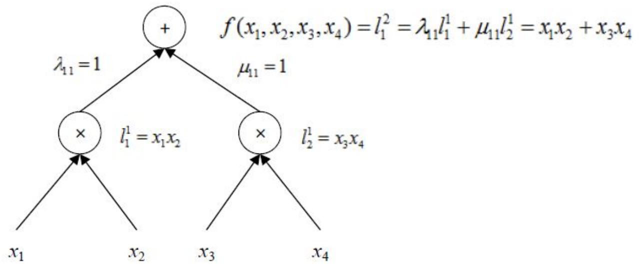


Figure: A sum-product networks with $n = 4$

Representational ability of deep structured model

- The output of the sum-product network is given by $f(x_1, ..., x_n) = l_1^{2i} \in \mathbb{R}$.
- The total number of nodes in the network is $1 + 2 + 4 + \cdots + 2^{2i-1} = n - 1$ for arbitrary i, the network size has a linear complexity.
- If we use the single-hidden-layer sum-product network to compute $f(x_1, ..., x_n)$, we have to modify it as a weighted sum of products of input variables.
- It can be shown that the number of products is $m_{2i} = 2^{\sqrt{n}-1}$, so the total number of units is $2^{\sqrt{n}-1} + 1$, so the networks size has an exponential complexity.

Representational ability of deep structured model

- Thus, deep networks may need fewer units and parameters for representing complex functions.
- Using a similar number of units, deep networks are generally more powerful to represent various functions than shallow networks.

Problems with deep structured model

Vanishing gradient problem

- As we add more and more hidden layers, back-propagation becomes less and less useful in passing information so the lower layers.
- In effect, as information is passed back, the gradients begin to vanish and become small relative to the weights of the networks.
- Therefore, prediction power of trained deep structured neural network is bad, even worse than power of shallow neural network.

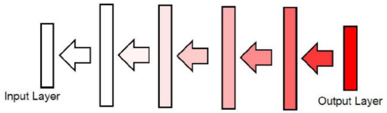


Figure: Vanishing gradient

Pre-training with unsupervised learning

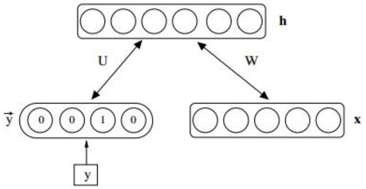
- A paper was published in *Science*, titled "Reducing the dimensionality of data with neural networks"(G.E.Hinton and R.R.Salakhutdinov, 2006), where RBMs are used to layer-wise pre-train autoencoders for overcoming the difficulties in training deep networks.
- The learned features can not only represent the nature of data, but also facilitate tasks of visualization and classification.

DBN with labeled data

- Let assume that the DBN with labeled data has l hidden layers.
- The contrastive divergence algorithm can be used to initialized each layer of a DBN as an RBM.
- When initializing the weights to $\mathbf{h}^{(l)}$, an RBM is trained to model the concatenation of y and $\mathbf{h}^{(l-1)}$:

$$P(\mathbf{x}, y, \mathbf{h}) \propto \exp(\mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{b}^T \mathbf{x} + \mathbf{c}^T \mathbf{h} + \mathbf{d}^T \vec{y} + \vec{y}^T \mathbf{U} \mathbf{h})$$

where $\vec{y} = (1_{y=i})_{i=1}^{C}$.(H.Larochelle and Y.Bengio, 2008)



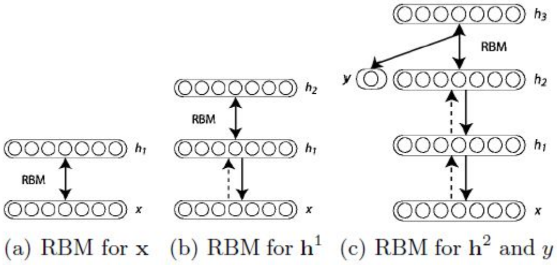
DBN with labeled data

- It is tractable to compute a representation $\mathbf{h}^{(l-1)}$ for the input by setting $\hat{\mathbf{h}}^{(0)} = \mathbf{x}$ and iteratively computing $\hat{\mathbf{h}}^{(k)} = P(\mathbf{h}^{(k)}|\mathbf{h}^{(k-1)})$ using RBM.
- Then, we can compute the probability of all classes given the approximately inferred value $\hat{\mathbf{h}}^{(l-1)}$ for $\mathbf{h}^{(l-1)}$ using the following expression:

$$P(y|\hat{\mathbf{h}}^{(l-1)}) = \sum_{\mathbf{h}^{(l)}} P(y, \mathbf{h}^{(l)}|\hat{\mathbf{h}}^{(l-1)})$$

- The network can then be fine-tuned by maximizing the log-likelihood of the class assignments using standard back-propagation algorithm.

DBN with labeled data



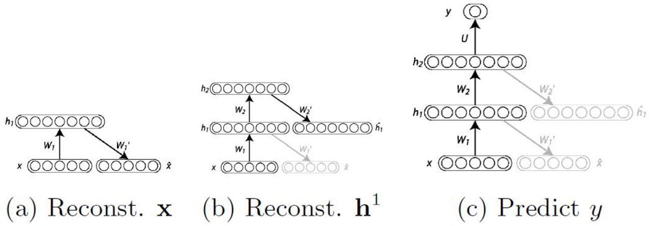
h_3
RBM
y
h_2
RBM
h_1
RBM
x
(a) RBM for $\mathbf{x}$
(b) RBM for $\mathbf{h}^1$
(c) RBM for $\mathbf{h}^2$ and y

Figure: Iterative pre-training construction of a DBN

SAE with labeled data

- Note that in AE, we compute output for reconstructing the input $\mathbf{x}$:
$$f(\mathbf{x}) = \sigma(\tilde{\mathbf{b}} + \tilde{\mathbf{W}}\sigma(\mathbf{b} + \mathbf{W}\mathbf{x}))$$
- Once an AE is trained, its internal "bottleneck" representation (here, $\sigma(\mathbf{b} + \mathbf{W}\mathbf{x})$) can be used as the input for training a second AE and etc.
- The stacked AE can then be fine-tuned with respect to a supervised training criterion, using standard back-propagation algorithm.

SAE with labeled data



(a) Reconst. $\mathbf{x}$ (b) Reconst. $\mathbf{h}^1$ (c) Predict y

Figure: Iterative pre-training construction of SAE model

Application to the MNIST data

Experiments (H.Larochelle *el al.*, 2007)

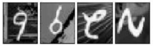
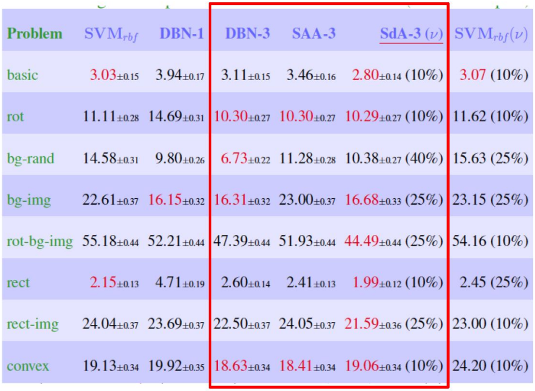
basic: subset of MNIST digits.	(10 000 training samples)
rot: applied random rotation (angle between 0 and 2π radians)	
bg-rand: background made of random pixels (value in $0 \dots 255$)	
bg-img: background is random patch from one of 20 images	
rot-bg-img: combination of rotation and background image	
rect: discriminate between tall and wide rectangles.	
rect-img: same but rectangles are random image patches	
convex: discriminate between convex and non-convex shapes.	

Figure: Data description

Application to the MNIST data

Experiments (H.Larochelle *el al.*, 2007)



Problem	SVM$_{rbf}$	DBN-1	DBN-3	SAA-3	SdA-3 (ν)	SVM$_{rbf}(\nu)$
basic	3.03$_{\pm 0.15}$	3.94$_{\pm 0.17}$	3.11$_{\pm 0.15}$	3.46$_{\pm 0.16}$	2.80$_{\pm 0.14}$ (10%)	3.07 (10%)
rot	11.11$_{\pm 0.28}$	14.69$_{\pm 0.31}$	10.30$_{\pm 0.27}$	10.30$_{\pm 0.27}$	10.29$_{\pm 0.27}$ (10%)	11.62 (10%)
bg-rand	14.58$_{\pm 0.31}$	9.80$_{\pm 0.26}$	6.73$_{\pm 0.22}$	11.28$_{\pm 0.28}$	10.38$_{\pm 0.27}$ (40%)	15.63 (25%)
bg-img	22.61$_{\pm 0.37}$	16.15$_{\pm 0.32}$	16.31$_{\pm 0.32}$	23.00$_{\pm 0.37}$	16.68$_{\pm 0.33}$ (25%)	23.15 (25%)
rot-bg-img	55.18$_{\pm 0.44}$	52.21$_{\pm 0.44}$	47.39$_{\pm 0.44}$	51.93$_{\pm 0.44}$	44.49$_{\pm 0.44}$ (25%)	54.16 (10%)
rect	2.15$_{\pm 0.13}$	4.71$_{\pm 0.19}$	2.60$_{\pm 0.14}$	2.41$_{\pm 0.13}$	1.99$_{\pm 0.12}$ (10%)	2.45 (25%)
rect-img	24.04$_{\pm 0.37}$	23.69$_{\pm 0.37}$	22.50$_{\pm 0.37}$	24.05$_{\pm 0.37}$	21.59$_{\pm 0.36}$ (25%)	23.00 (10%)
convex	19.13$_{\pm 0.34}$	19.92$_{\pm 0.35}$	18.63$_{\pm 0.34}$	18.41$_{\pm 0.34}$	19.06$_{\pm 0.34}$ (10%)	24.20 (10%)

Figure: Test errors on the datasets

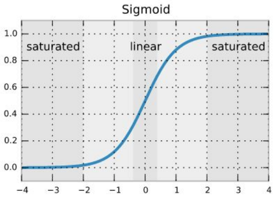
A new activation function(*ReLUs*)

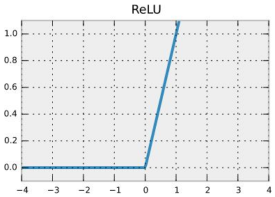
***ReLU* Nonlinearity**(V.Nair and G.E.Hinton, 2010)

- Rectified Linear Units
- It is just another activation function which is non-saturating nonlinearity

$$ReLU(s) = \max(0, s)$$

- CNN with *ReLU*s train several times faster than their equivalents with saturated units.
- And if we use *ReLU*s, we do not have to take it into consideration about vanishing gradient problem.





A new activation function

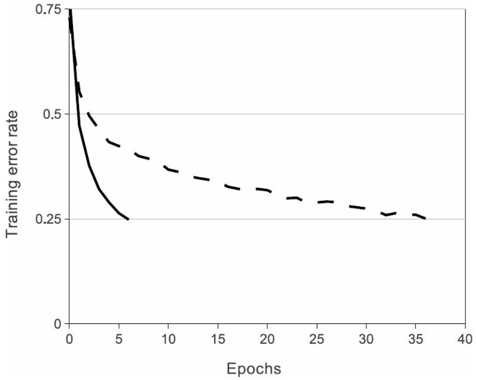


Figure: A 4-layer CNN with *ReLU*s(solid line) reaches a 25% training error rate 6 times faster than an equivalent networks with tanh neurons(dashed line).

Deep model for vision recognition

Convolutional neural networks(CNN)

- A CNN is primarily designed to deal with the recognition of 2D shapes inspired by the visual neural mechanism.
- In 1983, Fukushima proposed the model of *neocognitron*, which is considered to be the first implemented CNN, based on the receptive field.(K.Fukushima *et al.*, 1983)
- The standard design of CNN, *LeNet-5* network(Y.LeCun *et al.*, 1998), can classify digits successfully.
- The structure of standard CNN is composed of input layer, convolutional layer, pooling layer, convolutional layer, pooling layer,...,fully-connected layers and output layer.

Deep model for vision recognition

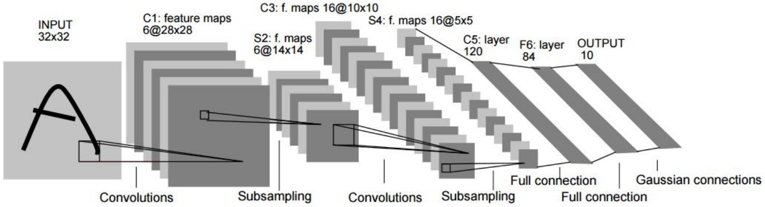


Figure: Architecture of *LeNet-5*

Deep model for vision recognition

Convolutional Layers

- Suppose that we have some $N \times N$ square neuron layer which is followed by our convolutional layer.
- If we use an $m \times m$ filter $\mathbf{w}$, our convolutional layer output will be of size $(N - m + 1) \times (N - m + 1)$.
- Wee sum up the contributions (weighted by the filter components) from the previous layer cells:

$$z_{ij}^{(l)} = \sum_{a=0}^{m-1} \sum_{a=0}^{m-1} w_{ab} h_{(i+a)(j+b)}^{(l-1)}$$

- Then, the convolutional layer applies its nonlinearity:

$$h_{ij}^{(l)} = \sigma(z_{ij}^{(l)})$$

Deep model for vision recognition

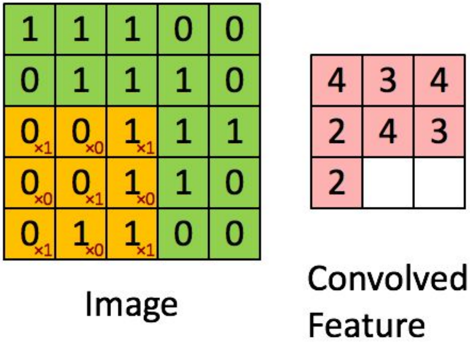


Figure: The convolution operation

Deep model for vision recognition

Max-pooling layers

- The max-pooling layers simply take some $k \times k$ region and output a single value, which is the maximum in that region.
- For instance, if their input layer is $N \times N$ layer, they will then output a $\frac{N}{k} \times \frac{N}{k}$ layer, as each $k \times k$ block is reduced to just a single value via a max function.

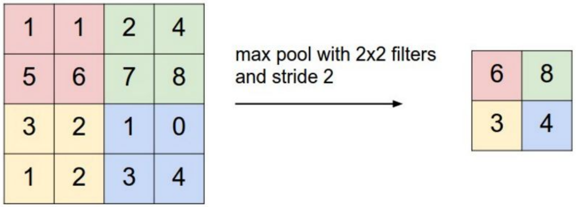


Figure: The max-pooling operation

Deep model for audio recognition

Recurrent neural networks(RNN)

- A recurrent neural network is a class of artificial neural network where connections between units form a directed cycle.
- Different from ordinary NN, a RNN can be used to analyze association relationships in time series.
- The computing procedure of a RNN from time 1 to T can be described as:

$$\begin{cases} \mathbf{h}(t+1) = \sigma(\mathbf{W}\mathbf{x}(t+1) + \mathbf{U}\mathbf{h}(t) + \mathbf{b}) & \text{where } 1 \leq t \leq T-1 \\ \mathbf{o}(t+1) = \sigma(\mathbf{V}\mathbf{h}(t+1) + \mathbf{c}) \end{cases}$$

Deep model for audio recognition

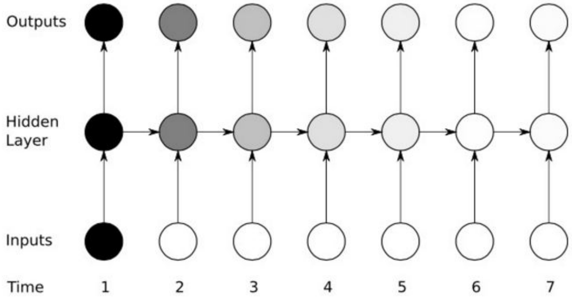


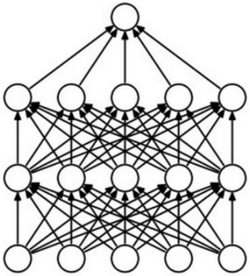
Figure: Architecture of *Vanilla* RNN

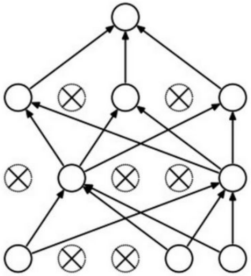
Miscellanea

Dropout(G.E.Hinton *et al.*, 2012)

- *Dropout* consists of setting to zero the output of each hidden neuron with probability 0.5.
- The neurons which are "dropped out" in this way do not contribute to the forward pass and do not participate in back-propagation.
- So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights.
- This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons.

Miscellanea


(a) Standard Neural Net


(b) After applying dropout.

Miscellanea

Local Response Normalization (A.Krizhevsky *et al.*, 2012)

- Let assume we consider about CNN.
- A.krizhevsky *el al.* found that the following local normalization scheme improves prediction ability.
- Denoting by $a^i_{x,y}$ the activity of a neuron computed by applying kernel i at position (x, y) and then applying the *ReLU* nonlinearity, the response-normalized activity $b^i_{x,y}$ is given by

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

where the sum runs over n "adjacent" filter maps at the same spatial position, and N is the total number of filters in the layer.

Miscellanea

- The constants k, n, α and β are hyper-parameters whose values are determined using a validation set.
- A.krizhevsky *el al.* used $k = 2, n = 5, \alpha = 10^{-4}$ and $\beta = 0.75$.

Miscellanea

Data Augmentation

- Let assume we consider about CNN.
- Data augmentation consists of altering the intensities of the RGB channels in training images.
- Specifically, perform PCA on the set of RGB pixel values throughout the training set, and to each training image, add multiples of the found PC with magnitudes proportional to the corresponding eigenvalues times a random variable drawn from a Gaussian with mean zero and standard deviation 0.1.

Miscellanea

- Therefore to each RGB image pixel $I_{xy} = (I_{xy}^R, I_{xy}^G, I_{xy}^B)^T$ we add the following quantity:

  $$[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

  where $\mathbf{p}_i$ and λ_i are ith eigenvector and eigenvalue of the 3×3 covariance matrix of RGB pixel values, respectively, and α_i is the random variable.

Application of new techniques

***AlexNet* (A.Krizhevsky *et al.*, 2012)**

- A deep CNN to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes.
- It uses *ReLU*, *dropout*, local response normalization and data augmentation techniques.
- It achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art.

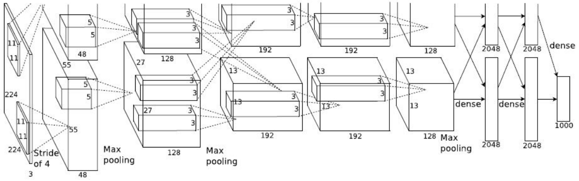


Figure: Architecture of *AlexNet*.

References

- O.Delalleau and Y.Bengio. Shallow vs. deep sum-product networks. *Advances in Neural Information Processing Systems*. pp.666-674. 2011.
- G.E.Hinton and R.R.Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*. 313(5786). pp.504-507. 2006.
- H.Larochelle and Y.Bengio. Classification using discriminative restricted Boltzmann machines. *Proceedings of the 25th international conference on Machine learning*. pp.536-543. 2008.
- K.Fukushima, S.Miyake and T.Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *Systems, Man and Cybernetics, IEEE Transactions on*. 5. pp.826-843. 1983

References

- H.Larochelle, D.Erhan, A.Courville, J.Bergstra and Y.Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. *Proceedings of the 24th international conference on Machine learning*. pp.473-480. 2007.
- Y.LeCun, L.Bottou, Y.Bengio and P.Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 86(11). pp.2278-2324. 1998.
- V.Nair and G.E.Hinton. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. pp.807-814. 2010.
- G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors.*arXiv preprint arXiv:1207.0580*. 2012.

References

- A.Krizhevsky, I.Sutskever and G.E.Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. pp.1097-1105. 2012.