18 Machine Learning

AI > ML > deep > DNN

supervised (regression, classification), unsupervised, reinforcement

neuroscience diverge, converge

Hasson, Uri, Samuel A. Nastase, and Ariel Goldstein. "Direct fit to nature: An evolutionary perspective on biological and artificial neural networks." Neuron 105, no. 3 (2020): 416-434.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the brain. Nature Reviews Neuroscience, 21(6), 335-346.

hierarchy missing until now

Hubel, David H., and Torsten N. Wiesel. "Receptive fields and functional architecture of monkey striate cortex." The Journal of physiology 195, no. 1 (1968): 215-243.

model neurons

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5, 115-133.

perceptrons

Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.

Minsky, Marvin, and Seymour A. Papert. Perceptrons: An introduction to computational geometry. MIT press, 2017.

XOR history

deep

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

linear vs nonlinear coefficients in functions

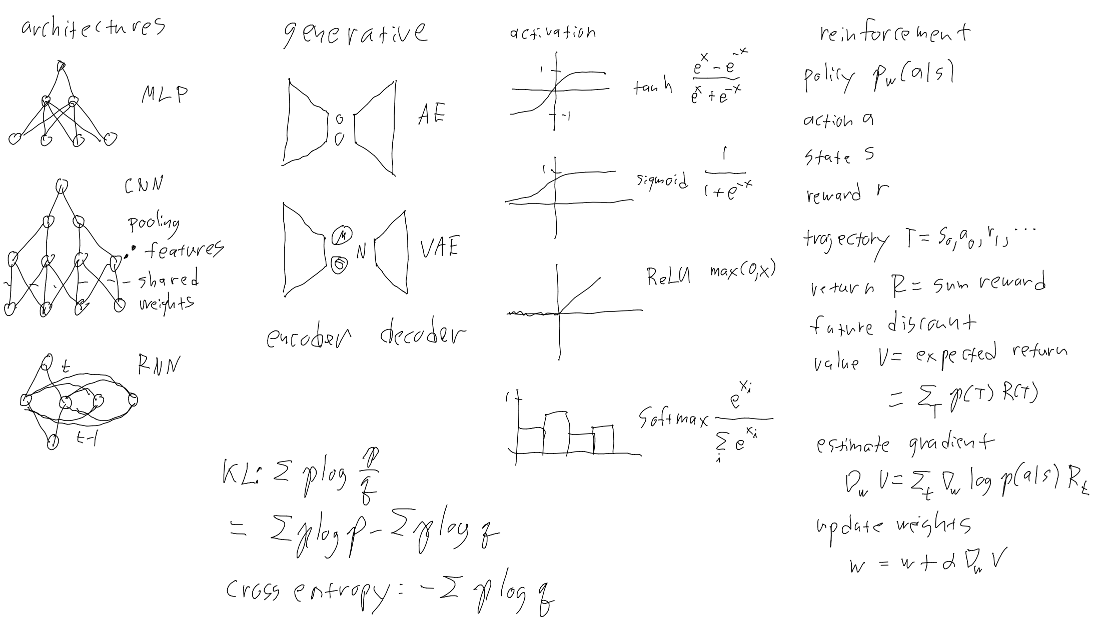
under mild assumptions linear depth vs exponential breadth

Telgarsky, M. (2016, June). Benefits of depth in neural networks. In Conference on learning theory (pp. 1517-1539). PMLR.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., & Liao, Q. (2017). Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. International Journal of Automation and Computing, 14(5), 503-519.

architecture, activation, training

tools appendix XOR example



architectures
MLP
CNN
pooling
features
shared weights
RNN
t
t-1
generative
AE
VAE
N
encoder decoder
activation
tanh $\frac{e^x - e^{-x}}{e^x + e^{-x}}$
sigmoid $\frac{1}{1+e^{-x}}$
ReLU max(0,x)
softmax $\frac{e^{x_i}}{\sum_i e^{x_i}}$
KL: $\sum p \log \frac{p}{q}$
$= \sum p \log p - \sum p \log q$
cross entropy: $-\sum p \log q$
reinforcement
policy $p_w(a|s)$
action a
state s
reward r
trajectory $T = s_0, a_0, r_1, \ldots$
return R = sum reward
future discount
value V = expected return
$= \sum_T p(T) R(T)$
estimate gradient
$\nabla_w V = \sum_t \nabla_w \log p(a|s) R_t$
update weights
$w = w + \alpha \nabla_w V$

Figure 18.1.

18.1 Architectures

18.1.1 Deep

MLP
- linear layers
- activation nodes
- transforms layers, FFT

18.1.2 Convolutional

pattern recognition
- invariance translation rotation
- convolutional
- *LeCun, Yann, Koray Kavukcuoglu, and Clément Farabet. "Convolutional networks and applications in vision." In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pp. 253-256. IEEE, 2010.*
- pooling

18.1.3 Recurrent

time
- MLP fixed window, FIR
- RNN, IIR

Pineda, Fernando J. "Generalization of back-propagation to recurrent neural networks." Physical review letters 59, no. 19 (1987): 2229.

unroll, backprop through time

vanishing, diverging gradients

LSTM, memory, HMM

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

18.1.4 Generative

autoencoder

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. AIChE journal, 37(2), 233-243.

encoder decoder

latent space

discontinuous

VAE

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

parameterize latent distribution

GAN

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks." Communications of the ACM 63, no. 11 (2020): 139-144.

attention, transformers

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

18.2 ACTIVATION FUNCTIONS

step function, logic switch

not differentiable

tanh

sigmoid

softmax

saturation

ReLU

Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." In Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807-814. 2010.

leaky ReLU

Xu, J., Li, Z., Du, B., Zhang, M., & Liu, J. (2020, July). Reluplex made more practical: Leaky ReLU. In 2020 IEEE Symposium on Computers and communications (ISCC) (pp. 1-7). IEEE.

GeLU

Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415.

18.3 TRAINING

pre-process data, zero mean unit variance, sphering, standardization

18.3.1 Loss

MSE, regression

cross entropy, KL distribution distance, classification

18.3.2 Backpropagation

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science.

inputs x_j

combine with weights

$$y_i = \sum_j w_{ij} x_j \tag{18.1}$$

can add bias for fixed values and to adjust sensitivity

$$y_i = \sum_j w_{ij} x_j + b_i \tag{18.2}$$

output through activation function

$$x_i = f(y_i) \tag{18.3}$$

hidden layer

$$\begin{aligned} x_i &= f\left[\sum_j w_{ij} f(y_j)\right] \\ x_i &= f\left[\sum_j w_{ij} f\left[\sum_k w_{jk} x_k\right]\right] \end{aligned} \tag{18.4}$$

hidden layers

$$x_i = f\left[\sum_j w_{ij} f\left[\sum_k w_{jk} f(y_l)\right]\right]$$

$$x_i = f\left[\sum_j w_{ij} f\left[\sum_k w_{jk} f\left[\sum_l w_{kl} x_l\right]\right]\right] \tag{18.5}$$

loss, data d

$$\chi^2 = \sum_n \sum_i \left[x_{i,n} - d_{i,n}\right]^2 \tag{18.6}$$

gradient descent, back-propagation

$$w_{ij} \to w_{ij} - \alpha \frac{\partial \chi^2}{\partial w_{ij}} \tag{18.7}$$

$$b_i \to b_i - \beta \frac{\partial \chi^2}{\partial b_i} \tag{18.8}$$

last layer

$$\begin{aligned}
\frac{\partial \chi^2}{\partial w_{ij}} &= \sum_n \sum_{i'} 2\left(x_{i',n} - d_{i',n}\right) \frac{\partial x_{i',n}}{\partial w_{ij}} \\
&= \sum_n 2\left(x_{i,n} - d_{i,n}\right) f'\left(y_{i,n}\right) x_{j,n} \\
&\equiv \sum_n \Delta_{i,n} x_{j,n}
\end{aligned} \tag{18.9}$$

$$\begin{aligned}
\frac{\partial \chi^2}{\partial b_i} &= \sum_n \sum_{i'} 2\left(x_{i',n} - d_{i',n}\right) \frac{\partial x_{i',n}}{\partial b_i} \\
&= \sum_n 2\left(x_{i,n} - d_{i,n}\right) f'\left(y_{i,n}\right) \\
&\equiv \sum_n \Delta_{i,n}
\end{aligned} \tag{18.10}$$

next layer

$$\begin{aligned}
\frac{\partial \chi^2}{\partial w_{jk}} &= \sum_n \sum_i 2\left(x_{i,n} - d_{i,n}\right) \frac{\partial x_{i,n}}{\partial w_{jk}} \\
&= \sum_n \sum_i 2\left(x_{i,n} - d_{i,n}\right) f'\left(y_{i,n}\right) w_{ij} f'\left(y_{j,n}\right) x_{k,n} \\
&= \sum_n \sum_i \Delta_{i,n} w_{ij} f'\left(y_{j,n}\right) x_{k,n} \\
&= \sum_n f'\left(y_{j,n}\right) \sum_i w_{ij} \Delta_{i,n} x_{k,n} \\
&\equiv \sum_n \Delta_{j,n} x_{k,n}
\end{aligned} \tag{18.11}$$

$$
\begin{aligned}
\frac{\partial \chi^2}{\partial b_j} &= \sum_n \sum_i 2\left(x_{i,n} - d_{i,n}\right) \frac{\partial x_{i,n}}{\partial b_j} \\
&= \sum_n \sum_i 2\left(x_{i,n} - d_{i,n}\right) f'\left(y_{i,n}\right) w_{ij} f'\left(y_{j,n}\right) \\
&= \sum_n \sum_i \Delta_{i,n} w_{ij} f'\left(y_{j,n}\right) \\
&= \sum_n f'\left(y_{j,n}\right) \sum_i w_{ij} \Delta_{i,n} \\
&\equiv \sum_n \Delta_{j,n}
\end{aligned} \tag{18.12}
$$

next layer

$$
\begin{aligned}
\frac{\partial \chi^2}{\partial w_{kl}} &= \sum_n \sum_i 2\left(x_{i,n} - d_{i,n}\right) \frac{\partial x_{i,n}}{\partial w_{kl}} \\
&= \sum_n \sum_i 2\left(x_{i,n} - d_{i,n}\right) f'\left(y_{i,n}\right) \sum_j w_{ij} f'\left(y_{j,n}\right) w_{jk} f'\left(y_{k,n}\right) x_{l,n} \\
&= \sum_n \sum_i \Delta_{i,n} \sum_j w_{ij} f'\left(y_{j,n}\right) w_{jk} f'\left(y_{k,n}\right) x_{l,n} \\
&= \sum_n \sum_j \Delta_{j,n} w_{jk} f'\left(y_{k,n}\right) x_{l,n} \\
&= \sum_n f'\left(y_{k,n}\right) \sum_j w_{jk} \Delta_{j,n} x_{l,n} \\
&\equiv \sum_n \Delta_{k,n} x_{l,n}
\end{aligned} \tag{18.13}
$$

$$
\begin{aligned}
\frac{\partial \chi^2}{\partial b_k} &= \sum_n \sum_i 2\left(x_{i,n} - d_{i,n}\right) \frac{\partial x_{i,n}}{\partial b_k} \\
&= \sum_n \sum_i 2\left(x_{i,n} - d_{i,n}\right) f'\left(y_{i,n}\right) \sum_j w_{ij} f'\left(y_{j,n}\right) w_{jk} f'\left(y_{k,n}\right) \\
&= \sum_n \sum_i \Delta_{i,n} \sum_j w_{ij} f'\left(y_{j,n}\right) w_{jk} f'\left(y_{k,n}\right) \\
&= \sum_n \sum_j \Delta_{j,n} w_{jk} f'\left(y_{k,n}\right) \\
&= \sum_n f'\left(y_{k,n}\right) \sum_j w_{jk} \Delta_{j,n} \\
&\equiv \sum_n \Delta_{k,n}
\end{aligned} \tag{18.14}
$$

forward, backward training passes
stochastic gradient descent

Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." In Proceedings of COMPSTAT'2010, pp. 177-186. Physica-Verlag HD, 2010.

incremental, batch updates for large data sets
learning rate, too slow, fast
momentum, local minima
ADAM adaptive rate, momentum
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

18.3.3 Regularization

hyperparameters
validation vs testing data
overfitting
early stopping
Caruana, Rich, Steve Lawrence, and Lee Giles. "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping." In NIPS, pp. 402-408. 2000.
weight decay, penalize sum square weights
Krogh, A., & Hertz, J. (1991). A simple weight decay can improve generalization. Advances in neural information processing systems, 4.
dropout, randomly drop weights
Wager, S., Wang, S., & Liang, P. S. (2013). Dropout training as adaptive regularization. Advances in neural information processing systems, 26.

18.3.4 Reinforcement

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play." Science 362, no. 6419 (2018): 1140-1144.
state, action, reward
policy
discount, return, value
policy gradient
learn policy directly
REINFORCE
Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Reinforcement learning, 5-32.
can be inefficient to train
learn value, derive policy
Q, DQN
Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).
can be inefficient for high-dimensional actions
many variants

18.4 SIMULATION

protein folding

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool et al. "Highly accurate protein structure prediction with AlphaFold." Nature 596, no. 7873 (2021): 583-589.

materials science

Choudhary, Kamal, Brian DeCost, Chi Chen, Anubhav Jain, Francesca Tavazza, Ryan Cohn, Cheol Woo Park et al. "Recent advances and applications of deep learning methods in materials science." npj Computational Materials 8, no. 1 (2022): 59.

Physics-Informed Neural Networks

PDEs

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics, 378, 686-707.

ODEs

K. Fricke, R. G. Nascimento, A. D. Marques, and F. A. C. Viana, Python Implementation of Ordinary Differential Equations Solvers using Hybrid Physics-informed Neural Networks, v0.0.1, Zenodo, https://github.com/PML-UCF/pinn_ode_tutorial, 10.5281/zenodo.3895408.

loss = MSE data + MSE governing equations

18.5 SELECTED REFERENCES

[Fleuret, 2023] Fleuret, F. (2023). *The Little Book of Deep Learning.* (preprint).

A lovely concise introduction.

[Ekman, 2021] Ekman, M. (2021). *Learning deep learning: Theory and practice of neural networks, computer vision, NLP, and transformers using Tensorflow..*

A good balance between breadth and depth.

18.6 PROBLEMS

(17.1) Train a multi-layer perceptron to predict the output of an order 10 maximal-length linear feedback shift register from the preceding 10 values, then generate new data by feeding the output back to the input and compare the generated and correct sequences.

(17.2) (*a*) *variant*: Train an autoencoder on data sets generated from $y = a + bx + cx^2$ by randomly varying a, b, and c, and evaluate its performance as a function of the size of the latent layer.

(*b*) *variant*: Train an autoencoder to find DTMF tones (Problem 12.5).

(17.3) Train a physics-informed neural network on a parametrically driven pendulum (Problem 6.4), and then compare the trajectory it generates with the correct one.

(17.4) (*a*) *variant*: Use policy gradient reinforcement learning to balance an inverted pendulum mounted on a cart driven by a controller that you train.

(*b*) *variant*: Use policy gradient reinforcement learning to learn how to control the angle of a parametrically driven pendulum.