Intro to Feedforward and Recurrent Neural Networks

Hands-on Deep Learning for Computer Vision WS16/17

Caner Hazırbaş

hazirbas@cs.tum.edu

Technische Universität München
Department of Computer Science
Computer Vision Group

October 18, 2016

Outline

Outline

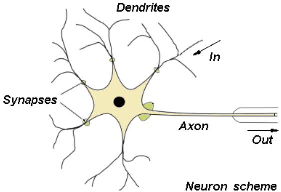
Introduction

- Neural networks are inspired by the biological neurons
  - The human brain (10^{10} cells) is the archetype of neural networks
- Most commonly used classifiers in machine learning
- Easily adaptable to regression and multi-class problems
- Representation learning method
- History
  - Computational model in 1943 [McCulloch and Pitts, 1943]
  - Backpropagation in 1975 [Werbos, 1974]
  - Neocognitron in 1980 [Fukushima, 1980]
  - Convolutional Neural Networks in 1998 [Lecun et al., 1998])
  - AlexNet in 2012 [Krizhevsky et al., 2012]
  - VGG-Net in 2014 [Simonyan and Zisserman, 2015]
  - ResNets: Deep Residual Networks [He et al., 2015]
    - ResNet-50, ResNet-101, and ResNet-152

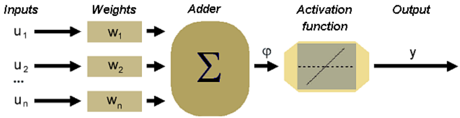
Outline

Perceptron

- One-neuron classifier
  - *linear* classifier
  - similar to SVMs
  - finds a hyperplane between classes
- Computational Model
- Neuron scheme [Gołda, 2005]



Neuron scheme



Neuron model

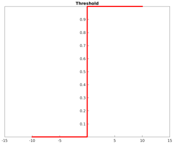
$$\varphi = \left(\sum_{i=1}^{n} w_i u_i + b\right)$$

$$y = \sigma(\varphi)$$

Activation Functions (σ)

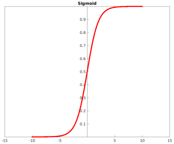
- Threshold Activation

$$\sigma(\varphi) = \begin{cases} 0 & \varphi \leq 0 \\ 1 & \varphi > 0 \end{cases}$$



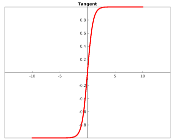
- Sigmoid Activation

$$\sigma(\varphi) = \frac{1}{1 + \exp^{-\varphi}}$$



- Tangent Activation

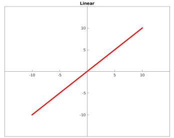
$$\sigma(\varphi) = \frac{\exp^{\varphi} - \exp^{-\varphi}}{\exp^{\varphi} + \exp^{-\varphi}}$$



Activation Functions (σ)

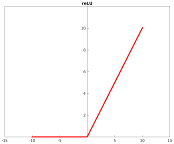
- Linear Activation

$$\sigma(\varphi) = \varphi$$



- Rectified Linear Unit (reLU)

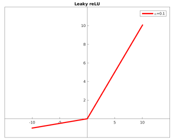
$$\sigma(\varphi) = \max(0, \varphi)$$



- Parametric reLU

$$\sigma(\varphi) = \max(\varphi, \alpha\,\varphi), \;\; \alpha < 1$$

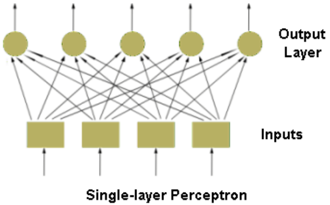
  - Leaky reLU when α is fixed



Outline

Single-layer Perceptron

- Single-layer Perceptron [Gołda, 2005]
  - multi-class classification
  - each output neuron is connected to each input neuron: ***fully-connected***



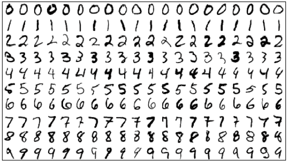
$$y_j = \sigma(\varphi_j) = \left(\sum_{i=1}^{n} \mathrm{w}_{ij}\, \mathrm{u}_i + b_j \right)$$

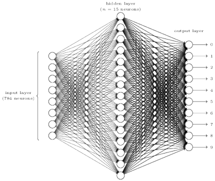
Multi-layer Perceptron

- Multilayer Perceptron
  - Stacked layers one after another
  - One or more hidden layers except input/output layers

$$y_j^{l+1} = \sigma(\varphi_j^l) = \left(\sum_1^n w_{ij}^l \, y_i^l + b_j^{l+1} \right)$$

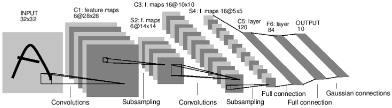
  - $l > 0$ is the layer index, $y_i^1 = u_i$
- MNIST Digit Classification with 2-layers neural networks [Nielsen, 2016]



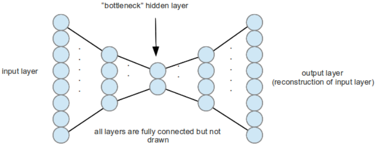


Applications of Neural Networks

- Classification [Lecun et al., 1998]

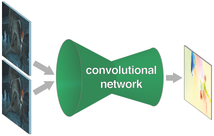


- Auto-encoders

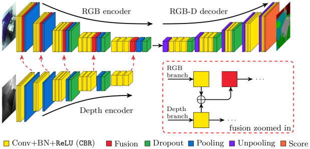


http://nghiaho.com/?p=1765

- Regression [Fischer et al., 2015]



- Transfer Learning [Hazirbas et al., 2016]



Outline

Gradient Descent

- Minimize a cost function. Let $\hat{y}(u)$ be the prediction and $y^*(u)$ be the expected output (ground-truth):

$$C(w,b) \equiv \frac{1}{2N}\sum_j^N ||\hat{y}(u_j) - y^*(u_j)||^2$$

- Highly non-convex respect to the parameter set (w,b). Thus no closed-form solution exist.
- Solution: Gradient Descent
  - Move in the opposite direction of the gradient
  - Let t be the iteration and η be learning rate, update rule for all w and b is then

$$w_{t+1} = w_t - \frac{\eta}{N}\sum_j^N \frac{\partial C_{U_j}}{\partial w_t}, \quad b_{t+1} = b_t - \frac{\eta}{N}\sum_j^N \frac{\partial C_{U_j}}{\partial b_t}$$

Backpropagation

- Minimize a cost function. Let $\hat{y}(u)$ be the prediction and $y^*(u)$ be the expected output (ground-truth):

$$C(w, b) \equiv \frac{1}{2N} \sum_{j}^{N} ||\hat{y}(u_j) - y^*(u_j)||^2$$

- Highly non-convex respect to the parameter set (w, b). Thus no closed-form solution exist.
- Backpropagation
  - forward pass all the inputs and compute the loss
  - propagate back the error through the layers
    - Take the derivative of the output of a layer w.r.t.its input and multiply with the error propagated down $\rightarrow$ ***chain-rule***
    - Update the parameters of the layer
  - repeat until convergence, *e.g.*, saturated-loss
- $\rightarrow$ example derivation: Exercise

Stochastic Gradient Descent (SGD)

- Gradient Descent
  - intractable for large datasets
  - high computational expense to compute the cost and derivatives for the entire dataset
  - not easily adaptable to 'online' setting
- Solution: SGD
  - compute the cost and derivatives over a batch of images
  - mini-batch ($m \ll N$) reduces the variance in the parameter update and can lead to more stable convergence

$$w_{t+1} = w_t - \frac{\eta}{m} \sum_{j}^{m} \frac{\partial C_{U_j}}{\partial w_t}, \quad b_{t+1} = b_t - \frac{\eta}{m} \sum_{j}^{m} \frac{\partial C_{U_j}}{\partial b_t}$$

  - use momentum for faster convergence

$$v_t \to v_{t+1} = \mu v_t + \frac{\eta}{m} \sum_{j}^{m} \frac{\partial C_{U_j}}{\partial w_t}, \quad w_t \to w_{t+1} = w_t + v_{t+1}$$

Other SGD-based Methods

- Adaptive Delta (AdaDelta)
- Adaptive Gradient (AdaGrad)
- Nesterovs' Accelerated Gradient (NAG)
- RMSProb
- Adaptive Moment Estimation (ADAM)
  - Less sensitive to initial learning rate and momentum

Back to Activation Functions

- Sigmoid Activation

$$\sigma(\varphi) = \frac{1}{1 + \text{exp}^{-\varphi}}$$

✗ Dying gradients (saturated activation)
✗ Non-zero centered

- Tangent Activation

$$\sigma(\varphi) = \frac{\text{exp}^{\varphi} - \text{exp}^{-\varphi}}{\text{exp}^{\varphi} + \text{exp}^{-\varphi}}$$

✗ Dying gradients (saturated activation)
✗ Non-zero centered

- Rectified Linear Unit (reLU)

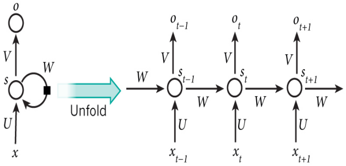
$$\sigma(\varphi) = \text{max}(0, \varphi)$$

✓ Greatly accelerated convergence
✓ Computationally cheap
✗ Can be fragile during training $\rightarrow$ use Leaky-reLU

Outline

Intro to RNNs

- Unfolding RNNs:



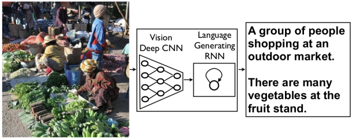
x_t is the input at time t

$s_t = \sigma(U \cdot x_t + W \cdot s_{t-1})$ hidden state at time t

o_t is the output at time t

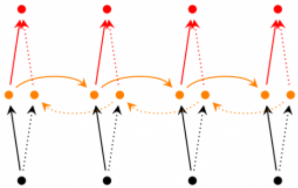
- $s_t \rightarrow$ *memory*
- ✗ long sequences
- ✓ use LSTMs.

- source: [Britz, 2015]
- for *sequential data*.
- Inputs (and outputs) are dependent.
  - *e.g.*next word in sentences.
- caption generation by [Vinyals et al., 2015]:



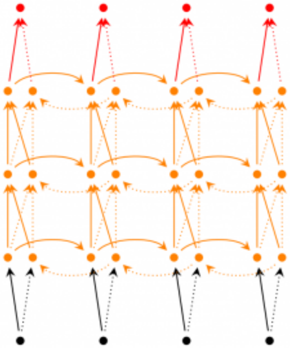
RNN extensions

- **Bidirectional RNNs**
  stacked RNNs.
  o_t depends both on o_{t-1} and o_{t+1}



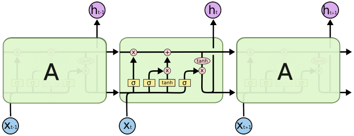
- **Gated Recurrent Unit (GRU)**
  varient of LSTM
  update gate as of forget+input gates

- **Deep (Bidirectional) RNNs**
  multilayer stacked RNNs.



RNN extensions

- Long Short Term Memory (LSTMs)



- Semantic segmentation [Byeon et al., 2015]

- source: [Olah, 2015]
- no vanishing gradient
- different activation function for the hidden state
- very efficient in practice

Back Propagation Through Time (BPTT)

- Gradient at each output depends on the current and previous steps.
- Propagate the gradient through time steps and sum up.
  - as same as unfolding th enetwork and then applying backpropagation
- *BPTT* has difficulty with local optima on RNNs.
  - Vanishing gradient problem.
- ***Conclusion***: no matter what, use ***LSTMs*** ✓

Lets play around...

`www.cs.stanford.edu/people/karpathy/convnetjs`

Quiz

Website

www.onlineted.com

Bibliography I

[Britz, 2015] Britz, D. (2015).
Introduction to rnns.
www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns.
Accessed: 2016-10-17.

[Byeon et al., 2015] Byeon, W., Breuel, T. M., Raue, F., and Liwicki, M. (2015).
Scene labeling with LSTM recurrent neural networks.
In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Fischer et al., 2015] Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., van der Smagt, P., Cremers, D., and Brox, T. (2015).
FlowNet: Learning Optical Flow with Convolutional Networks.
In *IEEE International Conference on Computer Vision (ICCV)*.

[Fukushima, 1980] Fukushima, K. (1980).
Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.
Biological Cybernetics, 36(4):193–202.

[Gołda, 2005] Gołda, A. (2005).
Introduction to neural networks.
http://home.agh.edu.pl/~vlsi/AI/intro/.

[Hazirbas et al., 2016] Hazirbas, C., Ma, L., Domokos, C., and Cremers, D. (2016).
Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture.
In *Asian Conference on Computer Vision (ACCV)*.

Bibliography II

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015).
Deep residual learning for image recognition.
arXiv preprint arXiv:1512.03385.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).
Imagenet classification with deep convolutional neural networks.
In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943).
A logical calculus of the ideas immanent in nervous activity.
The bulletin of mathematical biophysics, 5(4):115–133.

[Nielsen, 2016] Nielsen, M. (2016).
Neural networks and deep learning.
`http://neuralnetworksanddeeplearning.com`.

[Olah, 2015] Olah, C. (2015).
Understanding lstm networks.
`http://colah.github.io/posts/2015-08-Understanding-LSTMs`.
Accessed: 2016-10-17.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015).
Very deep convolutional networks for large-scale image recognition.
ICLR, abs/1409.1556.

Bibliography III

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015).
Show and tell: A neural image caption generator.
arXiv preprint arXiv:1411.4555.

[Werbos, 1974] Werbos, P. J. (1974).
Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.
PhD thesis, Harvard University.