Deep Learning and Temporal Data Processing

1 - Deep Neural Networks

Andrea Palazzi

June 21th, 2017

University of Modena and Reggio Emilia

Agenda

Introduction

Linear classifiers

Neural Networks

Training a DNN

Credits

References

Introduction

Deep Neural Networks: overview

[1]

Linear classifiers

Linear Classifier

For the purpose of this lecture, we'll stick to the task of image classification.

Let's assume we have a training dataset of N images

$$x_i \in \mathbb{R}^D, i = 1, \ldots, N$$

that we want to classify into K distinct classes.

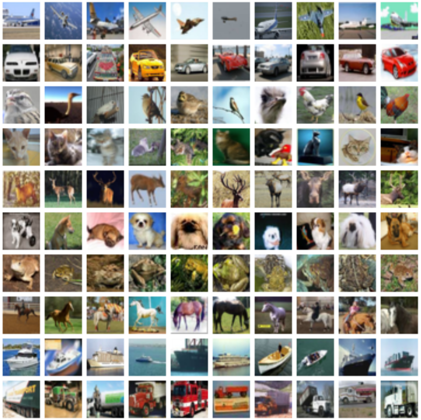
Thus, training set is made by couples:

$$(x_i, y_i), \ \ where \, y_i \in \{1, \ldots, K\}$$

Our goal is to define a function $f : \mathbb{R}^D \mapsto \mathbb{R}^K$ that maps images to class scores.

Linear Classifier

Making a real-world example: let's take the `CIFAR-10` dataset, which consists of $N = 60000$ 32x32 RGB images belonging to 10 different classes.



Linear Classifier

Each image is 32x32x3, thus it can be thought as a column vector $x_i \in \mathbb{R}^{3072}$.
Now we can define a **linear mapping**:

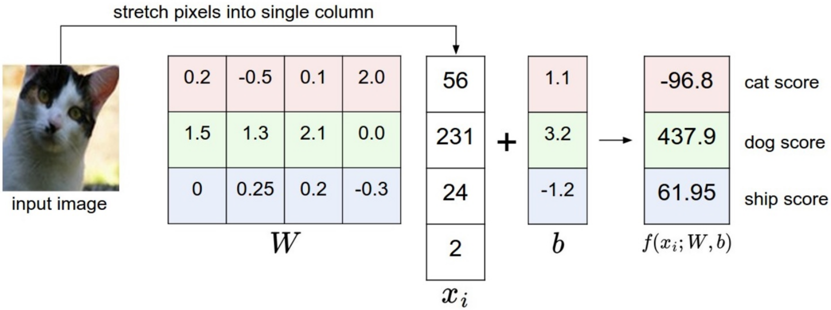
$$f(x_i, W, b) = Wx_i + b$$

where the parameters are:

- the weight matrix $W \in \mathbb{R}^{10x3072}$
- the bias vector $b \in \mathbb{R}^{10}$.

Intuitively, our goal is to learn the parameters from the training set *s.t.* when a new test image x_i^{test} is given as input, the score of the correct class is higher that the scores of other classes.

Linear Classifier



stretch pixels into single column

input image

$$\underset{W}{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}} \underset{x_i}{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}} + \underset{b}{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}} \longrightarrow \underset{f(x_i; W, b)}{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}} \begin{array}{l} \text{cat score} \\ \text{dog score} \\ \text{ship score} \end{array}$$

Example of mapping an image to a score. For the sake of visualization, here the image is assumed to have only 4 grayscale pixels.

Logistic Regression

Softmax Classifier

First let's introduce the **softmax function**:

$$softmax_j(\mathbf{z}) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

It takes a vector of arbitrary real-valued scores **z** and squashes it to a vector of values between zero and one that sum to one.

e.g.

$$\mathbf{z} = \begin{bmatrix} 1.2 \\ 5.1 \\ 2.7 \end{bmatrix} \quad softmax(\mathbf{z}) = \begin{bmatrix} 0.018 \\ 0.90 \\ 0.08 \end{bmatrix}$$

Softmax Classifier

Softmax Classifier generalizes Logistic Regression classifier to multi-class classification.

In the Softmax classifier the scores of linear function mapping $f(x_i, W) = Wx_i$ are interpreted as unnormalized log probabilities and we use the **cross-entropy loss**:

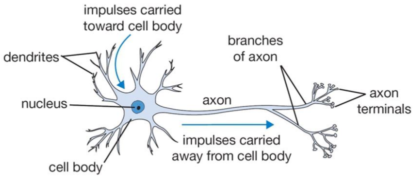
$$L_i = -log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Prove yourself that this loss function makes sense.

Neural Networks

Introduction

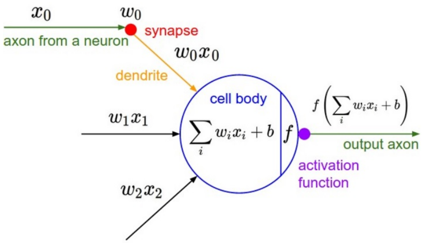
Neural Networks are a mathematical model **coarsely** inspired by the way our own brain works.


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

Neurons are the basic computational unit of our brain. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14} - 10^{15} **synapses**. Each neuron receives input signals from its **dendrites** and produces output signals along its (single) **axon**. The axon eventually branches out and connects via synapses to dendrites of other neurons.

Modeling a Single Neuron

More formally, we can model a single **neuron** as follows:



Each neuron can have multiple inputs. The neuron's output is the dot product between the inputs and its weights, plus the bias: then, a non-linearity is applied.

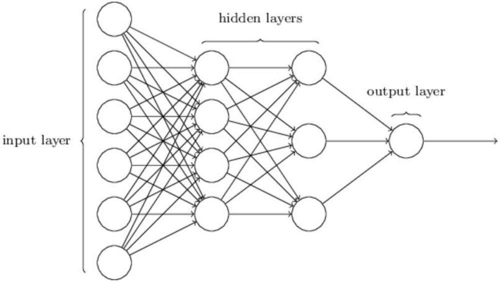
Modeling a Single Neuron

It's easy to see that a single neuron can be used to implement a binary classifier.

Indeed, when **cross-entropy loss** is applied to neuron's output, we can optimize a **binary Softmax classifier** (*a.k.a.* Logistic regression).

Neural Networks

When we connect an ensemble of neurons in an acyclic graph is when the magic happens and we get an actual **neural network**.



Neural networks are arranged in **layers**, with one *input layer*, one *output layer* and *N hidden layers* in the middle.

The network depicted here has a total of 47 *learnable parameters. Does this make sense to you?*

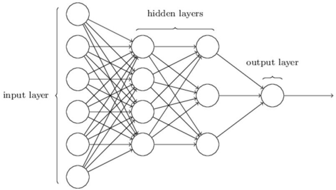
Neural Networks

The 4-layer network previously depicted can be simply expressed as:

$$out = \phi(\mathbf{W_3}\phi(\mathbf{W_2}\phi(\mathbf{W_1}\mathbf{x}))) \tag{1}$$

where:

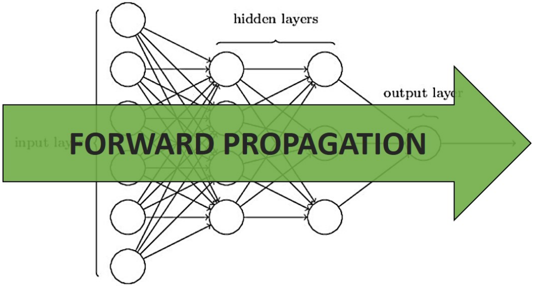
- ϕ is the activation function
- $\mathbf{x} \in \mathbb{R}^6$ is the input
- $\mathbf{W_1} \in \mathbb{R}^{4x6}$ are the weights of first layer
- $\mathbf{W_2} \in \mathbb{R}^{3x4}$ are the weights of second layer
- $\mathbf{W_3} \in \mathbb{R}^{1x3}$ are the weights of third layer



Notice that to ease the notation biases have been incorporated into weight matrices W.

Forward propagation

Forward propagation is the process of computing the network output given its input.



The formula of forward propagation for our toy network above is the one in Eq. 1.

Deep Neural Networks: why now

Training a DNN

Backpropagation

Credits

Credits

These slides heavily borrow from the following Stanford course:

- http://cs231n.stanford.edu/

if you want to deepen these concepts, please start from here!

Also, nice convolution animations are taken from here:

- https://github.com/vdumoulin/conv_arithmetic

References

References

[1] F. Yu and V. Koltun.
Multi-scale context aggregation by dilated convolutions.
arXiv preprint arXiv:1511.07122, 2015.