Course Notes: Deep Learning for Visual Computing

Peter Wonka

August 30, 2021

Contents

1 Fully Connected Neural Networks

1.1 Literature

- Andrew NG, coursera, course on machine learning, lecture about neural networks
- CS231n: Convolutional Neural Networks for Visual Recognition, Justin Johnson, Serena Yeung, Fei-Fei Li, youtube

1.2 NN Compute Complex Functions

- A neural network computes a function of an input tensor and maps it to an output tensor
- A neural network de nes a complex function that maps an input to an output
  - high-dimensional input
  - many parameters
  - possibly high-dimensional output

1.3 Simpli cation: Vectors

- Simpli cation: assume inputs (and outputs) are vectors (for now)
- Notation:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$

- We are dealing with
  - images, videos, volumes
  - point clouds
  - graphs, meshes
  - text - not so much in this class
- n is quite large

1.4 Function Review

- A function with n inputs and m outputs is denoted by $f : R^n \to R^m$
- Example: image classification
  - vectorized images as input
  - vector of class probabilities as ouput
  - $f : R^{1000000} \to R^{1000}$
- Linear function $f : R^n \to R$:

$$f(x) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + \text{constant}$$

  - Note: we do not distinguish linear / affine

1.5 Function Notation

- A function has variables and parameters and constants
- Different notations to distinguish between variables (x) and parameters (w)
  - Ignore parameters: $f(x) = w_1 x_1 + \ldots$
  - Use a semicolon: $f(x;w) = w_1 x_1 + \ldots$
  - Use a subscript: $f_w(x) = w_1 x_1 + \ldots$

1.6 Constant O set in Linear Functions

- Problem: For a linear function it would be much nicer to write $f(x;w) = w^T x$
- Solution: implicitly add an element 1 to every vector x

$$x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \quad w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{pmatrix}$$

$$f(x;w) = w^T x = w_0 + w_1 x_1 + \dots w_n x_n$$

- It is no longer clear what version is used (with or without 1).
- Typically, authors explicitly consider the o set w_0 in simple equations / gures, but omit o sets in complex equations / gures

1.7 Non-linear Function: Sigmoid

- Two popular options for non-linear functions are Sigmoid and ReLU
- Sigmoid: $sigmoid(x) = \sigma(x) = \frac{1}{1+e^{-x}}$

  - $sigmoid(10) \approx 0.9999$ and $sigmoid(-10) \approx 0.0001$

1.8 Non-linear Function: ReLU

- ReLU: $\mathrm{ReLU}(x) = \max(x, 0)$

- Reminder: vector version $\max(x, 0)$ computes the $\max$ elementwise

1.9 Building Complex Functions with Simple Building Blocks

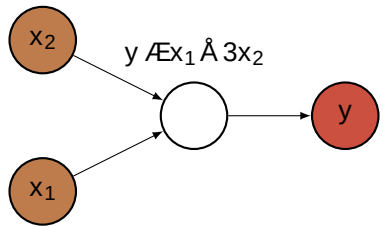
- We can use simple building blocks to build a complex function
- A building block is a simpler function, e.g. $y = f_i(x)$ or $Y = f_i(X)$
- Countless building blocks have been proposed
- One important criteria is that the building block has to be differentiable

1.10 Computational Graph

- A NN is a type of computational graph
- A Computational Graph is a directed graph that has nodes that are operations (function) or variables,
  - Data travels on the edges

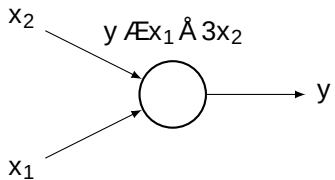
1.11 Computational Graph Drawing Choices

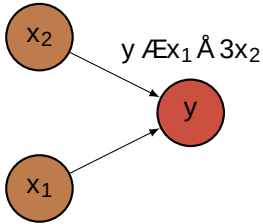
- Example: $f : R^2 ! R, y Æ f(x), y Æ x_1 Å 3x_2$
- Unusual drawing where output variable is drawn as a separate node



1.12 Computational Graph Drawing Choices

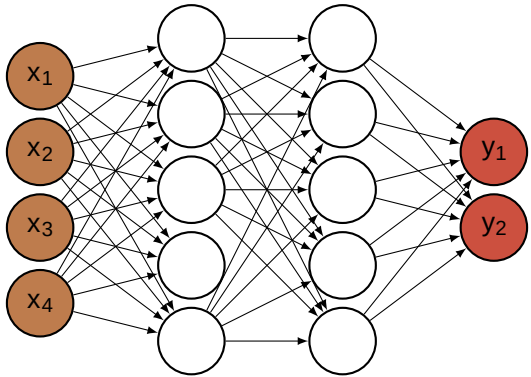
- Drawing variables as separate node or not
  - Typically only input variables are drawn as nodes
  - Variables can just be drawn on edges
  - Output node is typically not drawn at all
  - Confusing either way




x2
y ÆX1 Å 3x2
y
x1

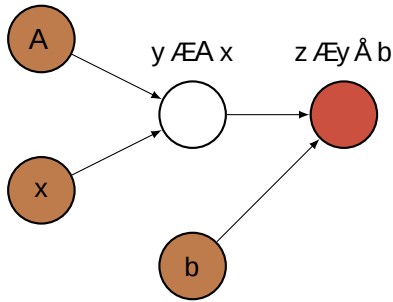
1.13 Scalar vs. Tensor Nodes

- Traditional deep learning: each node computes a single scalar value
- Overkill for drawing, manual computation, and programming
  - E.g. $f : R^4 ! R^2$ with two hidden layers



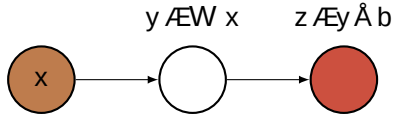
1.14 Computational Graph Drawing Choices

- Much easier with inputs and outputs as tensors
- Examplez Æƒƒƒƒƒƒƒf (A ,x,b)



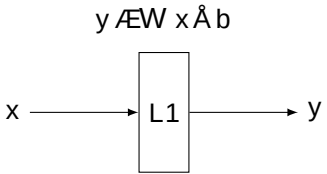
1.15 Different Types of Variables

- There are two types of variables
  - Example: $z = f(x; W, b) = Wx + b$
  - Input variables that describe the input data: x,
  - Parameters or weights of the neural network: W, b
  - Constants are fixed once
- Runtime: new input data determines variables: x; W, b are not changed.
- Training: input data x it taken from the training set; W, b are variables we optimize
- Simplification: not drawing network parameters



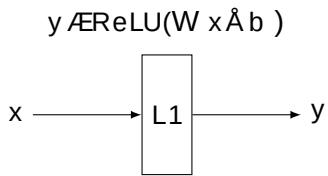
1.16 Layers

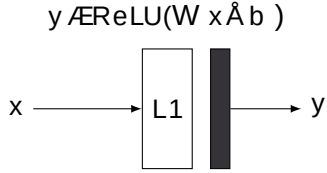
- A computational node is also called a layer of the network
- How complex should a layer be? The complete network could be a single layer
- Linear Layer: computes $y = linear(x; W, b) = Wx + b$
- A layer is often drawn as a rectangle (the height and width of the rectangle can b proportional to the size of the output tensor)

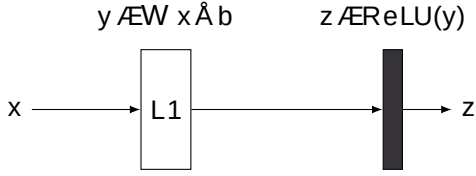


1.17 Non-linear Layers

- Non-linear layers work element-wise, e.g. ReLU(x) or sigmod(x)
- Traditional term: activation function
- Programming: separate layer that is implemented, e.g. on top of a linear layer
- Visualization: omitted, or drawn directly next to another layer




y ÆReLU(W x Å b)
x
L1
y

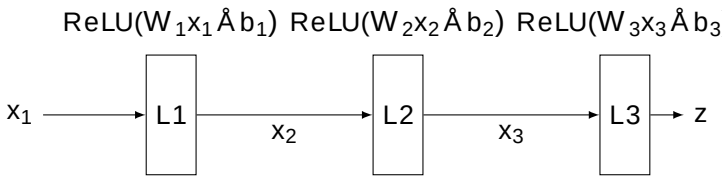

y ÆW x Å b
z ÆReLU(y)
x
L1
z

1.18 Batch Processing

- The neural network needs to process data in batches (mini-batches)
- If the input to a layer is a single vector x, a sequence of vectors can be stored as a matrix (rank-2 tensor)
- The layer has to process this rank-2 tensor X
  - Data can be in columns or rows
  - Careful with transpose! E.g. $\mathrm{ReLU}(WX + b)$? $\mathrm{ReLU}(X^T W + b^T)$?
  - Careful with broadcasting! E.g. $WX + b$. Is b added to columns or rows?
  - Broadcasting is not allowed in mathematics, only in computer science

1.19 Fully Connected Neural Network

- Fully Connected Neural Network : A NN that consists of alternating linear layers and non-linear layers
- Alternative term: Multi-layer Perceptron
- As nal layer, there is typically another type of non-linear layer, e.g. softmax for classi cation



1.20 Types of Neural Networks

- Feedforward Neural Networks vs. Recurrent Neural Networks:
  - Feedforward Neural Networks are also called Deep Feedforward Networks
  - Feedforward networks have no feedback connections.
  - Recurrent Neural Networks (RNNs) have feedback connections.
- Types of blocks
  - A Fully Connected Network also called Multi-layer Perceptron (MLP) uses (only) fully connected layers (and non-linear layers)
  - A fully connected layer is also called linear layer in PyTorch
  - A Convolutional Neural Network (CNN) uses predominantly convolutional layers

1.21 Neural Networks Alternative

- Traditionally, many people built neural networks using neurons that output a single scalar value
- We now look at this alternative variation

1.22 Building a single Neuron

- Neuron combines a linear function with a nonlinear function
- Neuron has many inputs, but only one output
- Example:

$$\frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

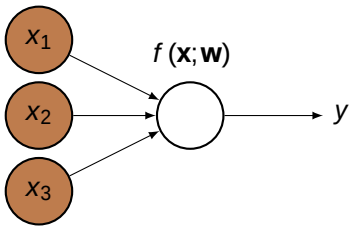
$$max(0, \mathbf{w}^T \mathbf{x})$$

- Terms: the non-linear function is called activation function

1.23 Single Neuron Drawing

- Example:

$$\begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \boxed{neuron} \rightarrow y = f(\mathbf{x};\mathbf{w})$$



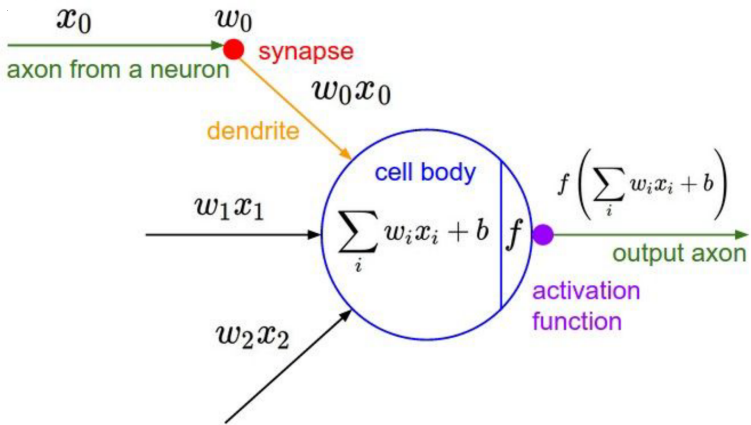
- Variables

$$\mathbf{x} = \begin{pmatrix} x_0 = 1 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

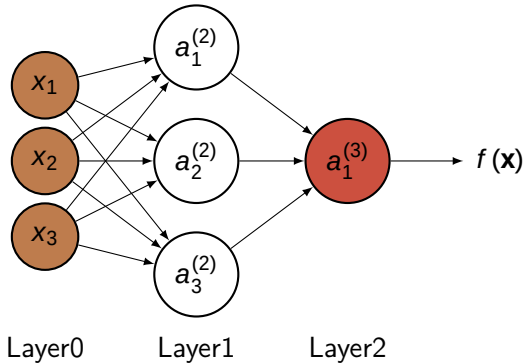
- The bias unit x_0 can be drawn as separate input or not
- The variables **w** are called weights or parameters of the network

1.24 Single Neuron Drawing Alternative

- Drawings inspired by transfer from neurology
  - taken from another context with slightly different variables



1.25 Building a Mini Network



- Terms:
  - Layer 0 is also called Input Layer
  - Layer 1 is a Hidden Layer

- Layer 2 is the Output Layer
- $a_i^{(j)}$ = activation of unit i in layer j
- $W^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

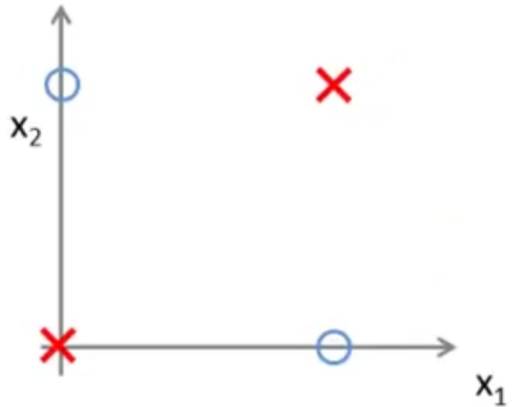
- Computation
  - $a_1^{(2)} = g(w_{10}^{(1)} x_0 + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3)$
  - $a_2^{(2)} = g(w_{20}^{(1)} x_0 + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3)$
  - $a_3^{(2)} = g(w_{30}^{(1)} x_0 + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3)$
  - $output = a_1^{(3)} = g(w_{10}^{(2)} a_0^{(2)} + w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)})$
- If network has s_j units in layer j and s_{j+1} units in layer $j+1$, then $W^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.
- In the example above, $W^{(1)}$ has $3 \times 4 = 12$ entries.
- Terms:
  - The inputs x_i could also be considered as activiations $a_i^{(1)}$

- Since all activations from layer j connect to all neurons of layer $j Å 1$ this network is called a fully connected neural network.

- Vectorized Implementation:
  - $\mathbf{a}^{(1)} Æ \mathbf{x}$
  - add $a_0^{(1)} Æ 1$
  - $lin^{(2)} Æ W^{(1)}\mathbf{a}^{(1)}$
  - $\mathbf{a}^{(2)} Æ nl(lin^{(2)})$
  - add $a_0^{(2)} Æ 1$
  - $lin^{(3)} Æ W^{(2)}\mathbf{a}^{(2)}$
  - $h_w(\mathbf{x}) Æ \mathbf{a}^{(3)} Æ nl(lin^{(3)})$

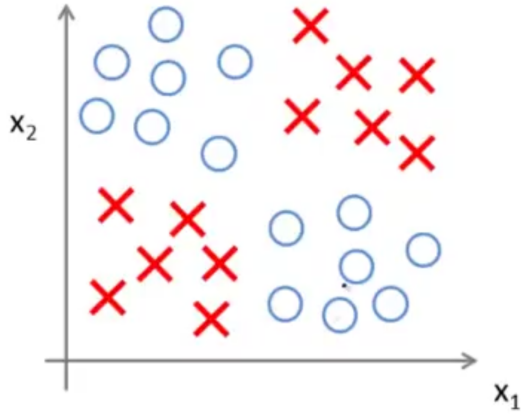
1.26 Computing Boolean Functions

- We are interested in computing Boolean functions as examples



- This example shows the XNOR function
- $y \text{Æ} x_1 XNOR x_2$
- $y \text{Æ} NOT(x_1 XOR x_2)$
- $x_1, x_2 \, 2 \{0,1\}$
- red X means $y \text{Æ} 1$, blue circle means $y \text{Æ} 0$
- Problem: there is no 2 layer (input and output layer) network that can build the XNOR function

- This could be imagined as a simplified version of a more complex classification example


x_2
x_1

1.26.1 Building the AND function

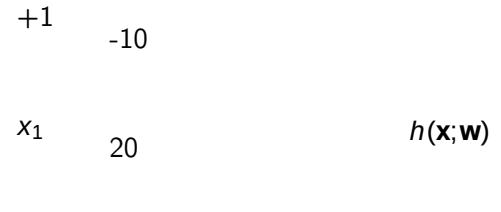
- $y = x_1 \, AND \, x_2$

- Verification:
  - reminder: $sigmoid(10) \approx 0.9999$ and $sigmoid(-10) \approx 0.0001$
  - $0 \, AND \, 0 = 0 \cdot 20 + 0 \cdot 20 + 1 \cdot (-30) = -30 < 0$

- 1 *AND* 0 Æ 20 ¡ 30 Æ ¡ 10 ¼ 0
- 0 *AND* 1 ¼ 0
- 1 *AND* 1 ¼ 1

1.26.2 Building the OR function

- $y = x_1 OR x_2$



- Verification:
  - reminder: $sigmoid(10) \approx 0.9999$ and $sigmoid(-10) \approx 0.0001$
  - $0 OR 0 = 0$

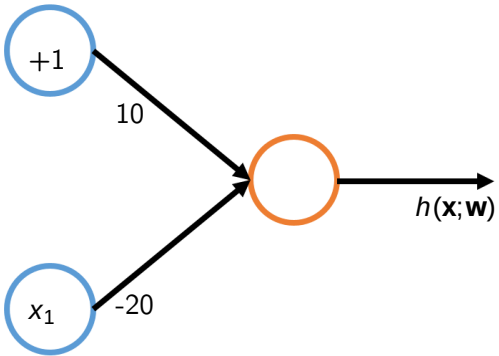
± 1*OR* 0 ¼ 1

± 0*OR* 1 ¼ 1

± 1*OR* 1 ¼ 1

1.26.3 Building the NOT function

- $y = NOT\ x_1$

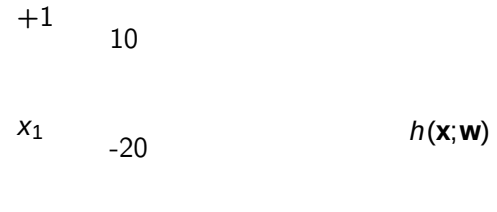


- Verification:
  - $NOT\ 0 = 1$

± *NOT* 1¼0

1.26.4 Building (NOT x_1) AND (NOT x_2)

- $y = (NOT\ x_1)\ AND\ (NOT\ x_2)$



- Verification:
  - $0\ op\ 0 = 1$
  - $1\ op\ 0 = 0$

- $0 op 1 ¼ 0$
- $1 op 1 ¼ 0$
- op is short for the operation $(NOT\, x_1)\, AND\, (NOT\, x_2)$