Deep learning and human vision

Mini lecture 3: review so far, and learning the weights

Deep neural network

Shallow neural network

Deep convolutional neural network

continuous valued inputs and outputs representing frequency of action potentials (spikes)

what determines the weights \( w_i \),?

Hubel & Wiesel, 1960s

Hubel & Wiesel, 1960s

receptive fields

a dictionary of image features?
hand-wired
shallow models

Models of weights based on a large body of empirical measurements characterizing the spatial filtering properties of neurons particularly in V1

What is convolution?

center-surround filter replaces each output cell with a weighted sum of the input levels

standard V1 model of "simple" cells:
including point-wise non-linearity

convolution

ReLU
Other non-linearities to model the output of an otherwise linear model of a neuron?

Table 1: Potential Computations That Can Be Performed by the Neural Circuits in Figure 1 at Their Steady States.

<table>
<thead>
<tr>
<th>Operation</th>
<th>(Steady-State) Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canonical</td>
<td>$y = \sum_{k=1}^{N} \frac{x_k}{\sum_{k=1}^{N} x_k}$ (2.1)</td>
</tr>
<tr>
<td>Energy model</td>
<td>$y = \sum_{k=1}^{N} x_k^2$ (2.2)</td>
</tr>
<tr>
<td>Sigmoid-like</td>
<td>$y = \sum_{k=1}^{N} \frac{x_k}{\sum_{k=1}^{N} x_k}$ (2.3)</td>
</tr>
<tr>
<td>Gaussian-like</td>
<td>$y = \sum_{k=1}^{N} \frac{x_k}{\sum_{k=1}^{N} x_k}$ (2.4)</td>
</tr>
<tr>
<td>Max-like</td>
<td>$y = \sum_{k=1}^{N} \frac{x_k}{\sum_{k=1}^{N} x_k}$ (2.5)</td>
</tr>
</tbody>
</table>


1-2 mm

V1 model of “simple” cells:

inclusion divisive normalization

linear

convolution

inputs from other nearby neurons—divisive normalization

an example of what DiCarlo et al. called a “normalized LN” mode, or NLN

Table 1: Potential Computations That Can Be Performed by the Neural Circuits in Figure 1 at Their Steady States.

<table>
<thead>
<tr>
<th>Operation</th>
<th>(Steady-State) Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canonical</td>
<td>$y = \sum_{k=1}^{N} \frac{x_k}{\sum_{k=1}^{N} x_k}$ (2.1)</td>
</tr>
<tr>
<td>Energy model</td>
<td>$y = \sum_{k=1}^{N} x_k^2$ (2.2)</td>
</tr>
<tr>
<td>Sigmoid-like</td>
<td>$y = \sum_{k=1}^{N} \frac{x_k}{\sum_{k=1}^{N} x_k}$ (2.3)</td>
</tr>
<tr>
<td>Gaussian-like</td>
<td>$y = \sum_{k=1}^{N} \frac{x_k}{\sum_{k=1}^{N} x_k}$ (2.4)</td>
</tr>
<tr>
<td>Max-like</td>
<td>$y = \sum_{k=1}^{N} \frac{x_k}{\sum_{k=1}^{N} x_k}$ (2.5)</td>
</tr>
</tbody>
</table>


Simple and complex cells in V1

Simple Cell

receptive field

Complex Cells

circuits for building translational invariance
shallow convolutional networks
what can they do?

detect edges

detect faces?  
(but not reliably)

textures?  
(with multiple channels  
or "features")

… but despite much research, few of these shallow networks work well as models of human perception, except for simple stimuli and tasks

e.g. they do work well as predictors of contrast detection, discrimination of fairly large family of textures.

Deep networks
what determines the weights $w_{ij}$ as one proceeds up levels ($j$) of the hierarchy?,

the tasks of vision,  
e.g. "core" recognition

the regularities in images,  
e.g. high correlations between nearby pixels

Deep solutions to the translational invariance problem

hierarchical models for feature extraction
given task constraints, e.g. core recognition

• Local features progressively grouped into more structured representations
  
• edges => contours => fragments => parts => objects

• Selectivity/invariance trade-off
  
• Increased selectivity for object/pattern type
  
• Decreased sensitivity to view-dependent variations of translation, scale and illumination
simple & complex cells in V1

- Simple cells
  - “template matching”, i.e. detect conjunctions, logical “AND”

- Complex cells
  - insensitivity to small changes in position, detect disjunctions, logical “OR”

- Recognition as the hierarchical detection of “disjunctions of conjunctions”

Recognize the letter “-htmlcode n"$t$n”

“htmlcode n"$t$n” is represented by the conjunction of a vertical and horizontal bar

\[ \text{AND} \rightarrow \text{htmlcode n"$t$n} \]

which can occur at any one of many locations $i$

\[ \text{“htmlcode n"$t$n”: } h_1 \& \& v_1 \parallel h_2 \& \& v_2 \parallel h_3 \& \& v_3 \ldots \]
learning the weights?

• instead of “hand wiring”, can the weights—the parameters required for subsequent inferences—be learned?
  
  • “machine learning”
  
  • two main approaches
  
  • unsupervised learning
  
  • supervised learning
shallow supervised learning

- Lots of algorithms for regression and classification
  - Linear regression
  - Simple perceptron
  - Fisher linear discriminant
  - Support vector machines
    - linear & non-linear

online training: iterate through the training pairs

https://datasciencelab.wordpress.com/2014/01/10/machine-learning-classics-the-perceptron/

non-linear support vector machines

Use a “kernel” to transform the input data into a higher dimensional space where the classes may be linearly separable.

http://www.slideshare.net/m80m07/support-vector-machine-15142742

deep supervised learning

training set: annotated datasets, with error back-propagation learning

general problems

- non-linear classification requires much trial and error pre-processing to find feature representations that enable linear separation of the classes
- over-fitting
  - too many free parameters relative to the number of training examples. Good fits to what has been learned, but poor prediction given new sample inputs
- the bias/variance trade-off
- solutions?
  - keep the model simple—e.g. “shallow”, fewer parameters to learn, but bigger errors
  - go for more parameters but then need more data AND better algorithms
deep supervised learning

training set: annotated datasets, with error back-propagation learning

next week
