Computing with High-Dimensional Vectors

Module 10
Relations to neural networks*

Denis Kleyko

*Lots of images from Internet were used to prepare this presentation
The only rule is, there are no rules:

- HD vectors as **input to neural networks**
- Neural networks for **producing HD vectors**
- HD Computing/VSA **connections** to **randomized** neural networks
- Use of HD Computing/VSA **primitives** in neural networks **design**
- **Explaining neural networks** with HD Computing/VSA
HD vectors as input to neural networks
Types of input

• HD vectors as a way to represent input to a network
  • Natural, as neural networks are also working with distributed representations

• Data to be fed to a neural network are high-dimensional and sparse
  • HD vectors can form more compact representation

• Input of varying size
  • Composite data structures
  • HD vectors are fixed size input

• Natural language processing
  • A lot of structure in language which can be potentially represented in HD vectors

• Expansion of the applicability of neural networks
  • Relieves the pressure of forming the task
    • With fixed size input
    • A sequence suitable for recurrent neural networks
Embedding $n$-gram statistics: Text classification

- Representation of $n$-gram statistics as before
- Text classification
  - 4 datasets
  - 9 ML algorithms
- Figures for neural network
  - 3 datasets

Composite data structures: Varying length sequences

- Trajectory is considered as a bag of n-grams
  - Trajectories are of variable length
- A tri-gram of locations \( \{l_1, l_2, l_3\} \) is represented as HD vector:
  - \( \rho^2(l_1) \oplus \rho^1(l_2) \oplus \rho^0(l_3) \)

Composite data structures: Vehicle Behavior Prediction

- HD vectors to encapsulate spatial information of multiple objects using the binding operation
  - Number of objects is a variable
- HD vectors as input to a LSTM for seq-to-seq prediction of vehicle positions
  - 5s into the future

\[ S_t = \text{TGRT} \otimes \text{TYPE}_{\text{target}} \otimes X^t \otimes Y^t \]

\[ \bigoplus_{\text{obj}} \text{TYPE}_{\text{obj}} \otimes X^{x_{\text{obj},t}} \otimes Y^{y_{\text{obj},t}} \]

- Best result in crowded and potentially dangerous driving situations
Composite data structures: Natural-to Formal-Language Generation

• Tensor Product Representations-based binding
  • Claim: the use of TPRs allows explicit capturing of relational structure to support reasoning

• Represent input data as superposition of tensors representing role-filler pairs
  • Structured representations of inputs are mapped to the structured representations of outputs

• Represent output data as tensor

• The model is not straightforward
  • But demonstrated to obtain good results on 2 datasets
    • MathQA
    • AlgoLisp

Neural networks for producing HD vectors
Types of output

• Transforming data to HD vectors might be a non-trivial task
  • Unstructured and of non-symbolic nature: images
  • Stimulates the interface between neural networks and HD computing/VSA in the other direction

• Transform activations of neural network layer(s) to HD vectors
  • Pre-trained convolutional neural networks
    • Increase the dimensionality
    • Change the format of representations
  • Purposefully train a network
    • Define cost function
Binary HD vectors from images

• Image hashing networks
  • Deep Quantization Network
  • Deep Cauchy Hashing Network
  • Deep Triplet Quantization Network

• Datasets
  • CIFAR-10
  • NUSWIDE_81

Aggregation of Image Descriptors

- Bunch of image descriptors
  - DELF
  - NetVLAD (NV)
  - AlexNet (AN)
  - DenseVLAD (DV)

- Place recognition datasets from mobile robotics
- Random projection controls dimensionality

- Form HD vector from holistic image descriptors:
  \[ H = \bigoplus_{i=1}^{k} H_i = \sum_{i=1}^{k} H_i \]

- Form HD vector from local image descriptors:
  \[ L = \bigoplus_{i=1}^{k} L_i \otimes P_i \]

Binary HD vectors from images

- Memory-augmented networks
  - Controller – Convolutional network
  - Key-value memory
    - Content addressable memory
    - Explicit memory
  - Few-shot learning

- Evaluation
  - Omniglot dataset
  - Phase-change memory devices

Binary HD vectors from images

- Memory-augmented networks
  - Controller – Convolutional network
  - Key-value memory
    - Content addressable memory
    - Explicit memory
  - Few-shot learning

- Evaluation
  - Omniglot dataset
  - Phase-change memory devices

HD Computing/VSA connections to randomized neural networks
Randomness in neural networks

- Stochastic assignment of a subset of the networks’ weights
  - Simpler (often linear) optimization problem

- Several broad families of models:
  - Randomized kernel Approximations
    - Chris’s Lecture for Module 8
  - Randomized feed-forward networks
  - Randomized connected recurrent networks

- Two fundamental ideas:
  - Randomization defines feature map lifting the input into a high-dimensional space
  - Resulting optimization problem is cast as a standard linear (regularized) least-squares

Randomly connected neural networks

Feed-forward neural networks:
- Random Vector Functional Link Networks, RVFL
- Extreme Learning Machines, ELM

Recurrent neural networks:
- Echo State Networks, ESN
- Liquid State Machines, LSM
Random Vector Functional Link Networks

- Three layers:
  - input
  - hidden
  - output

- Random and fixed connections

- Non-linear activation function – $\tanh(x)$

- Readout connections: RLS

\[
W_{\text{out}} = \left( H^T H + \lambda I \right)^{-1} H^T y
\]

Density-based encoding

- Connections input to hidden layers
  - Projection (lifting) to HD space
  - Random projection

- Transform scalars to HD vectors via density-based encoding
  - Thermometer codes
    - Chris’s Lecture for Module 8

- Binding operation with HD for weight and mapped scalar

0
1
2
3
N=4
Randomized neural networks via HD computing/VSA

• Weight matrix is random and bipolar
  • Interpreted as a set of HD vectors

• Binding operation with HD vectors for weight and mapped scalar
  • Associating each feature with its HD vector

• Bundling all associations -> linear activation of hidden layer

• Non-linear activation function -> bundling operation in a limited range
  • Clipping as a nonlinearity function

\[ \tanh(x) \]  
\[ \text{clipping}(x), k=3 \]
**intRVFL example**

- **Hidden layer (Reservoir) contains only integers in the limited range [-k, k]**
  - One neuron requires $\log_2(2k+1)$ bits -> $k=3$ – 3 bits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value, $x_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>0.276</td>
</tr>
<tr>
<td>#2</td>
<td>0.680</td>
</tr>
<tr>
<td>#3</td>
<td>0.955</td>
</tr>
<tr>
<td>#4</td>
<td>0.163</td>
</tr>
<tr>
<td>#5</td>
<td>0.119</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value, $v_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>3</td>
</tr>
<tr>
<td>#2</td>
<td>7</td>
</tr>
<tr>
<td>#3</td>
<td>10</td>
</tr>
<tr>
<td>#4</td>
<td>2</td>
</tr>
<tr>
<td>#5</td>
<td>1</td>
</tr>
</tbody>
</table>

**Quantized features, $\nu = [x_i, N], N=10$**

**Representation of quantized features via density-based encoding: $F; N=10$**

**Random weights (role vectors): $W^\text{in}; N=10$**

**Bound representations: $F \odot W^\text{in}$**

- **Input to hidden neurons: $\Sigma F \odot W^\text{in}$**
- **Non-linearity via clipping ($\kappa=2$), $h$**

- Determined by features
- Chosen randomly
- Activation of the hidden layer
intRVFL evaluation: classification on 121 datasets

- 121 datasets for classification from UCI Machine Learning Repository
- Number of examples: min – 10; max – 130064; median – 683;
- Number of features: min – 3; max – 262; median – 16;
- Number of classes: min – 2; max – 100; median – 3;

- Features were normalized to be in [0, 1] range

- Average accuracy of the best classifier (Random Forest): 0.82
- Average accuracy of the linear classifier: 0.73

Evaluation: results unlimited resources

- **Grid search:**
  - Number of hidden neurons (N) was varied in the range $[50, 1500]$ with step 50
  - Regularization parameter was in the range $2^{-10,5}$
  - $\kappa$ varied between $\{1, 3, 5, 7\}$

- **Floating point read-out matrix via RLS**

- **Average accuracy conventional RVFL:** 0.76

- **Average accuracy intRVFL:** 0.80

- **Extra experiments:**
  - RVFL with direct weights to input features: 0.76
  - RVFL with quantized features: 0.76
  - RVFL with optimized input projection: 0.71
  - RVFL with $N$ for proposed approach: 0.75
  - intRVFL with RVFL’s $N$: 0.78
Evaluation: results limited resources

- Fixed energy budget on FPGA
  - “Poorman’s” bounded optimality
  - Effectively it limits $n$

- Finite precision RVFL (8-bits)

- Average accuracy fixed point RVFL: \textbf{0.65}

- Average accuracy intRVFL: \textbf{0.73}

Echo State Networks

- An approach to Recurrent Neural Networks
- Three layers
  - input
  - hidden
  - output
- Non-linear activation function – \( \tanh(x) \)
- Random and fixed connections
- Recurrent connections between hidden neurons, \( \mathbf{W} \)

\[
x(n) = (1 - \alpha)x(n-1) + \alpha \tanh(\gamma \mathbf{W} x(n-1) + \beta \mathbf{W}^{\text{in}} \mathbf{u}(n))
\]

Integer Echo State Networks

- Reservoir contains only integers in the limited range \([-k, k]\)

- One neuron requires \(\log_2(2k+1)\) bits
  - \(k=3\) – 3 bits

\[
\text{tanh}(x) \quad \text{clipping}(x), \ k=3
\]

intESN evaluation: Time-series classification

- 800 neurons
- 3 datasets
- 3.9 times faster than ESN

<table>
<thead>
<tr>
<th>Multivariate datasets from UCI</th>
<th>#V</th>
<th>Train</th>
<th>Test</th>
<th>#C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character Trajectories</td>
<td>3</td>
<td>300</td>
<td>2558</td>
<td>20</td>
</tr>
<tr>
<td>Spoken Arabic Digit</td>
<td>13</td>
<td>6600</td>
<td>2200</td>
<td>10</td>
</tr>
<tr>
<td>Japanese Vowels</td>
<td>12</td>
<td>270</td>
<td>370</td>
<td>9</td>
</tr>
</tbody>
</table>
intESN evaluation: Time-series classification

- 800 neurons
- 3 datasets
- 3.9 times faster than ESN

Random vector for each class

• Neural networks can learn useful representation without modifying the weights of the output layer

\[
H_1 = \begin{bmatrix} 1 \\ \end{bmatrix}, \\
H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \\
H_3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, \\
H_4 = \begin{bmatrix} H & H \\ H & -H \end{bmatrix}
\]

• Hadamard matrix as a weight matrix

<table>
<thead>
<tr>
<th>Network</th>
<th>Dataset</th>
<th>Learned</th>
<th>Fixed</th>
<th># Params</th>
<th>% Fixed params</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet56 (He et al., 2016)</td>
<td>Cifar10</td>
<td>93.03%</td>
<td>93.14%</td>
<td>855,770</td>
<td>0.07%</td>
</tr>
<tr>
<td>DenseNet(k=12)(Huang et al., 2017)</td>
<td>Cifar100</td>
<td>77.73%</td>
<td>77.67%</td>
<td>800,032</td>
<td>4.2%</td>
</tr>
<tr>
<td>ResNet50 (He et al., 2016)</td>
<td>ImageNet</td>
<td>75.3%</td>
<td>75.3%</td>
<td>25,557,032</td>
<td>8.01%</td>
</tr>
<tr>
<td>DenseNet169(Huang et al., 2017)</td>
<td>ImageNet</td>
<td>76.2%</td>
<td>76%</td>
<td>14,149,480</td>
<td>11.76%</td>
</tr>
<tr>
<td>ShuffleNet(Zhang et al., 2017b)</td>
<td>ImageNet</td>
<td>65.9%</td>
<td>65.4%</td>
<td>1,826,555</td>
<td>52.56%</td>
</tr>
</tbody>
</table>

Learning next to nothing

- Fix fractions of convolutional layers of deep CNNs
- Allow only a small portion of the weights to be learned
- Performance can be on a par with learning all of them

Use of HD Computing/VSA primitives in neural networks design
Composite data structures: Natural-to Formal-Language Generation

- **Tensor Product Representations** based binding
  - Claim: the use of TPRs allows explicit capturing of relational structure to support reasoning.

- Represent input data as superposition of tensors representing role-filler pairs.
  - Both codebooks are learned.

- Represent output data as tensors.

- The model is pretty complicated.
  - But claimed to obtain good results on 2 datasets.

Binary HD vectors from images

- Memory-augmented networks
  - Controller – Convolutional network
  - Key-value memory
    - Content addressable memory
    - Explicit memory
  - Few-shot learning

- Evaluation
  - Omniglot dataset
  - Phase-change memory devices

Superposition of many neural networks into one

Problem(s):
- Online learning of multiple tasks
- Catastrophic forgetting
- Memory constrained environments

Approach:
- Use random HD vectors as context for tasks
- Use binding operation to associate with task’s set of parameters $W_k$
- Store models in superposition

Explaining neural networks with HD Computing/VSA
Qualitative: Binarized CNNs

• Why one can effectively capture the features in data with binary weights and activations?
  • Continuous vectors are well-approximated by binary vectors

• HD geometry
  • Binarization approximately preserves the direction of high-dimensional vectors
  • Weight-activation dot products are approximately proportional

Quantitative: Capacity theory

• What affects the accuracy?
  • $D, N, L$

• Can we predict retrieval accuracy?

\[ \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi \sigma_h}} e^{-\frac{(x-\mu_h)^2}{2\sigma_h^2}} (\Phi(x, \mu_r, \sigma_r))^D-1 \]

• Assumptions:
  • The distributions are normal
  • The distributions for “distractor” symbols are all the same
  • The distributions for different symbols are independent

2018 NeCo paper presented T1 describing the capacity of VSAs and some ESNs cases

Extensions of T1

• Need to extend the theory:

\[ \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \left( \Phi(x, \mu_j, \sigma_j) \right)^{D-1} \]

• T1:

• The distributions are normal
• The distributions for “distractor” symbols are all the same
• The distributions for different symbols are independent

• T2:

• The distributions are normal
• The distributions for “distractor” symbols are all the same
• The distributions for different symbols are independent

\[ \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \prod_{j=1, j \neq i}^{D-1} \Phi(x, \mu_j, \sigma_j) \]

• T3:

• The distributions are normal
• The distributions for “distractor” symbols are all the same
• The distributions for different symbols are independent

\[ \int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{x_1} dx_2 \cdots \int_{-\infty}^{x_1} dx_D \mathcal{N}(x, \mu, \Sigma) \]

From memory buffer to classification with neural networks

• Desire to get under the hood of neural networks

• Dissect the holistic functionality of neural network into two parts:
  • Multi-layer encoding stage $\rightarrow$ corresponds to $x$ in the memory buffer task
  • Single-layer classification by perceptron $\rightarrow$ similar to the regression-based perceptron in the memory buffer task

Pretrained deep networks on ImageNet with **T2**

**Normal distributions**

Correlation coefficient: 0.933

**Kernel distributions**

Correlation coefficient: 0.940
ImageNet subproblems on individual models

- We want to remove the bias
- Sub-problems of different size by randomly sampling from the ImageNet
- Prediction for sub-problems
- Bias for each model
ImageNet adjusted predicted accuracy

- Compensation for each network is based on the sub-problems on individual networks
- The compensations have almost removed bias and unsystematic deviations between the accuracies

Correlation coefficient: 0.998
ImageNet subproblems with T3

- Compensation needs to observe accuracies of smaller sub-problems

- Solution: independence assumption
  - Numerical integration is challenging
  - Sub-problems of size 4
Predictions via Monte Carlo sampling

• It is hard to calculate integral in $T_3$

• We try to estimate it by MC sampling
  • Correlation coefficient: 0.98

• Constant offset
  • No normalization constant
The only rule is, there are no rules:

- HD vectors as input to neural networks
- Neural networks for producing HD vectors
- HD Computing/VSA connections to randomized neural networks
- Use of HD Computing/VSA primitives in neural networks design
- Explaining neural networks with HD Computing/VSA
Relations to neural networks

Denis Kleyko