Convolutional and LSTM Neural Networks

Vanessa Jurtz

January 12, 2016
Contents

- Neural networks and GPUs
- Lasagne
- Peptide binding to MHC class II molecules
- Convolutional Neural Networks (CNN)
- Recurrent and LSTM Neural Networks
- Hyper-parameter optimization
- Training CNN and LSTM ensembles
Neural networks and GPUs

- A CPU (central processing unit) has few cores with lots of cache memory
- A GPU (graphics processing unit) has hundreds of cores but each with little memory
- GPUs are very good for lots of small operations that can be heavily parallelized, like matrix multiplication
- training a neural network requires lots of matrix multiplications!
Why use Lasagne?

- python
- it is highly flexible as you will see in the afternoon exercise
- code can be run on a CPU or GPU
- my collaborators are using it too
Peptide binding to MHC class II molecules

- MHC II only expressed in professional antigen presenting cells (dendritic cells, macrophages, B cells etc)
- only a small fraction of peptides bind MHC II and are potential T cell epitopes
- polygenic and highly polymorphic

Figure adapted from http://events.embo.org/12-antigen/
Peptides binding to MHCII molecules

Nielsen et al. 2010 "MHC Class II epitope predictive algorithms" Immunology

Why is this a hard problem?

- binding peptides have different lengths
- the binding core can lie anywhere within a peptide
- large number of different MHC class II molecules

Peptides binding HLA-DR3 adapted from Kenneth Murphy "Janeway’s Immunobiology" 8th edition fig 4.21 Garland Science
CNN - Convolutional Neural Networks

- convolutional filter has specified input size
- slides over input sequence feeding into a different hidden neuron in each position
- weights of the filter are fixed as it slides over input and updated in backpropagation
- usually multiple filters of the same and/or different size are used
Why could this be useful for predicting peptide binding to MHC class II?

- use filters of size 9 AA to focus on binding core
- integrate information about all 9-mers
CNNs are known to work well for image analysis

CIFAR example

http://deeplearning.net/tutorial/lenet.html
A biological example: subcellular localization of proteins

Sønderby et al. 2015 "Convolutional LSTM Networks for Subcellular Localization of Proteins"
http://arxiv.org/abs/1503.01919
Recurrent Neural Networks

- have connections between the neurons in the hidden layer
- have a memory "learn what to store and what to ignore" (Graves, 2015)

Figure 3.3: A recurrent neural network.

Graves 2015 "Supervised Sequence Labelling with Recurrent Neural Networks"
LSTM memory blocks

Instead of a hidden neuron we use an LSTM memory block:

Graves 2015 "Supervised Sequence Labelling with Recurrent Neural Networks"
LSTM Networks

Why make it more complicated?

LSTM memory blocks allow to store and access information over longer periods of time.

Graves 2015 "Supervised Sequence Labelling with Recurrent Neural Networks"
LSTM Networks

Input and output can be of variable length!

Example peptide MHC class II binding: many to one
LSTM Networks

- LSTM networks are known to work well for natural language processing
- check out Andrej Karpathys blog
- to read up: Graves 2015
LSTM Networks

Another biological application: protein secondary structure prediction

http://chemistry.berea.edu/~biochemistry/
Anthony/Aequorin/Aequorin3.php

Sønderby et al. 2015 "Convolutional LSTM Networks for Subcellular Localization of Proteins"
http://arxiv.org/abs/1503.01919
Hyper-parameter optimization

Parameters:
- weights

Hyper-parameters:
- weight initialization
- weight update
- activation function
- data encoding
- dropout
- learning rate
- number of CNN filters / number of LSTM memory blocks
- hidden layer size

Bergstra et al. 2012 "Random Search for Hyper-Parameter Optimization" Journal of Machine Learning Research
Hyper-parameter optimization - architectures

**CNN:**

- Peptide
  - Convolutional layer
    - MHC II pseudo sequence
  - Fully connected layer
    - Output

**LSTM:**

**A**

- Peptide
  - LSTM layer
  - LSTM layer
    - MHC II pseudo sequence
    - Fully connected layer
      - Output

**B**

- Peptide
  - LSTM layer
    - forward LSTM layer
    - backward LSTM layer
    - MHC II pseudo sequence
  - Fully connected layer
    - Output

**C**

- MHC II pseudo sequence + Peptide
  - LSTM layer
  - LSTM layer
    - Fully connected layer
      - Output

**D**

- MHC II pseudo sequence + Peptide
  - forward LSTM layer
  - backward LSTM layer
  - Fully connected layer
    - Output

- filter size varied (9 AA, 9+3 AA)
Hyper-parameter optimization - results

**CNN:**

Best performance per number of trials

**LSTM:**

Best performance per number of trials

single network performance (10% best nets for CNN + LSTM, entire ensemble of NetMHCIIpan-3.0)
CNN ensemble of 10 best hyper parameter settings + 5 seeds each

<table>
<thead>
<tr>
<th>Allele</th>
<th>#</th>
<th>#_bind</th>
<th>NetMHCIipan 3.0</th>
<th>CNN ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PCC</td>
<td>AUC</td>
<td>PCC</td>
</tr>
<tr>
<td>DRB1_0101</td>
<td>717</td>
<td>550</td>
<td>0.82</td>
<td>0.908</td>
</tr>
<tr>
<td>DRB1_0301</td>
<td>703</td>
<td>408</td>
<td>0.699</td>
<td>0.85</td>
</tr>
<tr>
<td>DRB1_0701</td>
<td>682</td>
<td>375</td>
<td>0.754</td>
<td>0.873</td>
</tr>
<tr>
<td>DRB1_0801</td>
<td>838</td>
<td>363</td>
<td>0.738</td>
<td>0.875</td>
</tr>
<tr>
<td>DRB1_1101</td>
<td>813</td>
<td>426</td>
<td>0.79</td>
<td>0.901</td>
</tr>
<tr>
<td>DRB1_1301</td>
<td>803</td>
<td>462</td>
<td>0.573</td>
<td>0.792</td>
</tr>
<tr>
<td>DRB1_1302</td>
<td>765</td>
<td>404</td>
<td>0.392</td>
<td>0.713</td>
</tr>
<tr>
<td>DRB1_1501</td>
<td>758</td>
<td>218</td>
<td>0.499</td>
<td>0.764</td>
</tr>
<tr>
<td>DRB3_0202</td>
<td>726</td>
<td>288</td>
<td>0.49</td>
<td>0.755</td>
</tr>
<tr>
<td>DRB3_0301</td>
<td>782</td>
<td>449</td>
<td>0.555</td>
<td>0.776</td>
</tr>
<tr>
<td>DRB4_0101</td>
<td>778</td>
<td>235</td>
<td>0.292</td>
<td>0.654</td>
</tr>
<tr>
<td>DRB4_0103</td>
<td>764</td>
<td>474</td>
<td>0.538</td>
<td>0.798</td>
</tr>
<tr>
<td>DRB5_0101</td>
<td>731</td>
<td>461</td>
<td>0.699</td>
<td>0.841</td>
</tr>
</tbody>
</table>

mean: 0.603 0.808 0.583 0.798
weighted mean: 0.602 0.807 0.583 0.798
Training CNN and LSTM ensembles

LSTM ensemble of 2 best hyper parameter settings + 2 seeds each

<table>
<thead>
<tr>
<th>Allele</th>
<th>#</th>
<th>#_bind</th>
<th>NetMHCIIpan 3.0</th>
<th>LSTM ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCC</td>
<td>AUC</td>
</tr>
<tr>
<td>DRB1_0101</td>
<td>717</td>
<td>550</td>
<td>0.820</td>
<td>0.908</td>
</tr>
<tr>
<td>DRB1_0301</td>
<td>703</td>
<td>408</td>
<td>0.699</td>
<td>0.850</td>
</tr>
<tr>
<td>DRB1_0701</td>
<td>682</td>
<td>375</td>
<td>0.754</td>
<td>0.873</td>
</tr>
<tr>
<td>DRB1_0801</td>
<td>838</td>
<td>363</td>
<td>0.738</td>
<td>0.875</td>
</tr>
<tr>
<td>DRB1_1101</td>
<td>813</td>
<td>426</td>
<td>0.790</td>
<td>0.901</td>
</tr>
<tr>
<td>DRB1_1301</td>
<td>803</td>
<td>462</td>
<td>0.573</td>
<td>0.792</td>
</tr>
<tr>
<td>DRB1_1302</td>
<td>765</td>
<td>404</td>
<td>0.392</td>
<td>0.713</td>
</tr>
<tr>
<td>DRB1_1501</td>
<td>758</td>
<td>218</td>
<td>0.499</td>
<td>0.764</td>
</tr>
<tr>
<td>DRB3_0202</td>
<td>726</td>
<td>288</td>
<td>0.490</td>
<td>0.755</td>
</tr>
<tr>
<td>DRB3_0301</td>
<td>782</td>
<td>449</td>
<td>0.555</td>
<td>0.776</td>
</tr>
<tr>
<td>DRB4_0101</td>
<td>778</td>
<td>235</td>
<td>0.292</td>
<td>0.654</td>
</tr>
<tr>
<td>DRB4_0103</td>
<td>764</td>
<td>474</td>
<td>0.538</td>
<td>0.798</td>
</tr>
<tr>
<td>DRB5_0101</td>
<td>731</td>
<td>461</td>
<td>0.699</td>
<td>0.841</td>
</tr>
</tbody>
</table>

mean: 0.603, 0.808, 0.605, 0.809
weighted mean: 0.602, 0.807, 0.605, 0.810
Future work

- combine CNN and LSTM in one network
- make an ensemble of different network architectures: CNN, LSTM, feed forward
- try to visualize what the networks learn
- try to find a way to extract/visualize the binding core
Aknowledgements

- Casper Sønderby
- Søren Sønderby
- Ole Winther
- Morten Nielsen