Deep Neural Networks

Beyond Feed-Forward Neural Networks

Alesandro Montemurro
PhD Student – Immunoinformatics and Machine Learning Group
almon@dtu.dk
Feed-forward Neural Networks (FNN)
When FNNs work – An example

• You want to predict the price of houses in Boston
• You have a dataset where, for each house you have certain features, e.g., squared meters, number of rooms, energetic class, ecc..
When FNNs do not work

• Sequential or time-dependent data

• Images analysis
Convolutional Neural Networks (CNN)

• Developed for images, but they work with other types of input
• Able to handle inputs with different sizes
• Able to extract the relevant features from the input by themselves
Convolutional Neural Networks

- Features extraction
- Convolution + pooling layers
- Max pooling

- Classification
- Fully connected layers
- Nx binary classification

- Convolution + nonlinearity

Example image of a bird with labeled features and classification output.
• Visualization of the filters
A more concrete example: peptide-MHC binding

- We don’t know what are the optimal features for this classification
- A FNN can work well if the signal is coherent across input peptides
- We can apply a 1D CNN to solve this problem
pep-MHC: FNN vs. CNN

• HLA-A:02*01
Most of the peptides are 9-mers with strong P2 and P9 binding anchors

⇒ A FNN can easily capture the signal, because P2 and P9 “do not move”

• HLA-A:03*01
The peptides length is in the range [8,11]

⇒ A FNN will most likely fail

⇒ A CNN can capture the signal thanks to the filters that slide along the peptide
Sequence Encoding

- The amino acid sequences are converted to numbers based on their biochemical profile (BLOSUM)
- Each AA is represented by a vector with 20 entries
- Example:
  A 9-mer is encoded as a [9x20] matrix
In practice

```python
class CNNpep(nn.Module):
    def __init__(self, n_filters, k, n_l1):
        super(CNNpep, self).__init__()
        self.conv_layer = nn.Conv1d(in_channels=21,
                                   out_channels=n_filters,
                                   kernel_size=k,
                                   stride=1,
                                   padding=0)

        self.fc1 = nn.Linear(n_filters, n_l1)
        self.fc2 = nn.Linear(n_l1, 1)
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        # The input dimensions are [batch_size, pep_len, channels]
        # PyTorch wants [batch_size, channels, pep_len]
        # So we swap the second and third dimension
        x = x.permute(0, 2, 1)
        x = self.relu(self.conv_layer(x))
        x, _ = torch.max(x, axis=2)
        x = self.relu(self.fc1(x))
        out = self.fc2(x)

        return out
```

NOTE: We are using a 1D convolution with 20 channels (coming from the BLOSUM encoding)

Global Max-Pooling: For each filter, we take the maximum value

Today’s exercise

• FNN on A0201-restricted peptides
• FNN on A0301-restricted peptides (look at the peptide length distribution)
• CNN on A0301-restricted peptides
It’s time to play!