How to train your DragoNN (Deep RegulAtory GenOmic Neural Network)

A workshop on Deep Learning for Regulatory Genomics

June 9th 2016
ENCODE Users Meeting
Stanford University
The dragonn package implements deep neural networks (DNNs) for regulatory genomics, methods for DNN interpretation, and provides tutorials showcasing dragonn models using sequence simulations.

For code, tutorial, and upcoming workshops: [http://kundajelab.github.io/dragonn/](http://kundajelab.github.io/dragonn/)
Primer with guidelines and in-vivo models coming soon!
Before we begin ..

Logging in and starting the tutorial
1. Point your browser to

http://mitra.stanford.edu/dragonn.html

This should bring up a login page to the dragonn client:

2. Your username is of the format lastnameFirstname based on the information you used to register for this tutorial.

The password is dragonn
Gene Regulation

Transcription factors (Regulatory proteins)

Promoter

Motif

Enhancer

Insulator

Protein

mRNA

Active Gene

Repessed Gene

DNA

GAAC TATT C

Repessed Gene

Repressed Gene

Transcription factors (Regulatory proteins)

Promoter

Motif

Enhancer

Insulator

Protein

mRNA

Active Gene

Repessed Gene

DNA
Sequence motifs

Set of aligned sequences

ATTATAGCAAACCTA
AACATGCCAAAGTA
ATCATCCAAAAGGA
ATCGTCCGAAAAGGA
AACGAGCGAAAAGGA

Position-specific scoring matrix

\[ p_i(x_i = a_i) \]

For a subsequence \( S = a_1, a_2, \ldots, a_k \) where \( a_i \in \{A, C, G, T\} \)

\[
\text{log-odds score}(S) = \sum_{i=1}^{k} \log_2 \left( \frac{p_i(x_i = a_i)}{p_{bg}(x_i = a_i)} \right) \geq \text{threshold} \Rightarrow \text{true hit}
\]

One-hot encoding (X)

Input sequence

G C A T T T A C C G A T A A
Learning regulatory sequence patterns

- Positive class of genomic sequences bound a TF of interest

Can we learn patterns (motifs) in the DNA sequence that distinguish these 2 classes of genomic sequences?

- Negative class of genomic sequences not bound by TF of interest
Simulation

Positive class of genomic sequences containing two motifs with relatively **fixed** spacing

Negative class of genomic sequences containing two motifs with **random** spacing
Supervised machine learning

Training
Input sequences (X)

Class = +1
Class = +1
Class = +1

Training
Output labels (Y)
Class = -1
Class = -1
Class = -1

'**Training**' means learning the function F(X) from multiple input, output pairs (X,Y)
Supervised machine learning

Test Input sequences \((X)\)

Classification function \(F(X)\)

Is its class \(Y = +1\) or \(-1\)?
Probability \((Y = +1 \text{ given } X)\)

‘Testing’ means predicting the output labels (class) \(Y\) for a new sequence \(X\) not used in training.
A simple classifier
(An artificial neuron)

\[ Y = F(x_1, x_2, x_3) \]

\[ Z = w_1.x_1 + w_2.x_2 + w_3.x_3 + b \]

\[ Y = h(Z) \]

Training means learning the optimal \( w \)'s and \( b \)

Logistic / Sigmoid

\[ h(Z) \]

ReLu (Rectified Linear Unit)

\[ h(Z) \]
Biological motivation of DCNN

Scan sequence using filters

Convolutional layers learn motifs (PSSM)

Threshold scores using ReLU

Max pool thresholded scores over windows

Predict probabilities using logistic neuron

Class label: Positive or Negative
Deep convolutional neural network

Logistic output

Typically followed by one or more fully connected layers

Maxpooling layers take the max over sets of conv layer outputs

Later conv layers operate on outputs of previous conv layers

Convolutional layer (same color = shared weights)

Maxpooling layer
- pool width = 2
- stride = 1

Conv Layer 2
- Kernel width = 3
- stride = 1
- num filters = 2
- total neurons = 6

Conv Layer 1
- Kernel width = 4
- stride = 2*
- num filters = 3
- Total neurons = 15

Maxpooling layers take the max over sets of conv layer outputs

Later conv layers operate on outputs of previous conv layers

Convolutional layer (same color = shared weights)

Maxpooling layer
- pool width = 2
- stride = 1

Conv Layer 2
- Kernel width = 3
- stride = 1
- num filters = 2
- total neurons = 6

Conv Layer 1
- Kernel width = 4
- stride = 2*
- num filters = 3
- Total neurons = 15

One-hot encoding (X)

Input sequence

G C A T T A C C G A T A A
Training a neural network

Learning weights via optimization algorithm called **stochastic gradient descent**

**Optimization Objective:** Minimize error (loss) on the training datasets
i.e. difference between true $Y$ and predicted output $F(X)$

- An incremental algorithm:
  - Present examples $(x_i, y_i)$ one at a time,
  - Modify $w$ slightly to increase the log-probability of observed $y_i$:

\[
  w := w + \eta \frac{\partial}{\partial w} \log p(y_i | x_i; w)
\]

where the learning rate $\eta$ determines how "slightly".
Measures of performance

TP, FP, FN, TN are absolute counts of true positives, false positives, false negatives and true negatives

- $N$ - sample size
- $N^+ = FN + TP$ number of positive examples
- $N^- = FP + TN$ number of negative examples
- $O^+ = TP + FP$ number of positive predictions
- $O^- = FN + TN$ number of negative predictions

<table>
<thead>
<tr>
<th>outputs \ labeling</th>
<th>$y = +1$</th>
<th>$y = -1$</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = +1$</td>
<td>TP</td>
<td>FP</td>
<td>$O^+$</td>
</tr>
<tr>
<td>$f(x) = -1$</td>
<td>FN</td>
<td>TN</td>
<td>$O^-$</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>$N^+$</td>
<td>$N^-$</td>
<td>$N$</td>
</tr>
</tbody>
</table>
Measures of performance

Area under the Receiver operating curve (auROC)
Compares sensitivity (recall) to false positive rate (1-specificity) at various thresholds
auROC = 1 (Perfect classifier)
auROC = 0.5 (Random classifier)
Measures of performance

Area under Precision-Recall Curve (auPRC)
Compares precision (1 - false discovery rate) to recall (sensitivity) at various thresholds
auPRC = 1 (Perfect classifier)

Positive predictive value (PPV), Precision
\[ \text{PPV} = \frac{\sum \text{True positive}}{\sum \text{Test outcome positive}} \]

True positive rate (TPR), Sensitivity, Recall
\[ \text{TPR} = \frac{\sum \text{True positive}}{\sum \text{Condition positive}} \]
Interpretation: In-silico mutagenesis

Output: Bound (+1) vs. not bound (0)

Assess change in output

Input: One-hot encoded DNA sequence
Interpretation: DeepLIFT (Deep learning feature importance)

P(Y=+1 given X)

Starting the tutorial

Logging in and starting the tutorial
1. Point your browser to

http://mitra.stanford.edu/dragonn.html

This should bring up a login page to the dragonn client:

2. Your username is of the format lastname_firstname based on the information you used to register for this tutorial.

The password is dragonn
Starting the tutorial

3. Click on the `workshoputorial.ipynb` link inside the examples folder to open up the Jupyter notebook for the tutorial.
4. Click the “Run All” in the “Cell” dropdown menu
# Sequence Simulations

In [2]: `print_available_simulations()`

<table>
<thead>
<tr>
<th>Simulation Name</th>
<th>“Positive” class sequence</th>
<th>“Negative” class sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>simulate_single_motif_detection</code></td>
<td>Contains a single motif</td>
<td>Random sequence</td>
</tr>
<tr>
<td><code>simulate_motif_counting</code></td>
<td>Contains many instances of a motif</td>
<td>Contains few instances of a motif</td>
</tr>
<tr>
<td><code>simulate_motif_density_localization</code></td>
<td>Contains multiple instances of a motif in center</td>
<td>Contains multiple instances of a motif throughout</td>
</tr>
<tr>
<td><code>simulate_multi_motif_embedding</code></td>
<td>Contains multiple motifs, one instance of each</td>
<td>Random sequence</td>
</tr>
<tr>
<td><code>simulate_differential_accessibility</code></td>
<td>Contains a group of motifs</td>
<td>Contains a different group of motifs</td>
</tr>
<tr>
<td><code>simulate_heterodimer_grammar</code></td>
<td>Contains two motifs positioned closely</td>
<td>Contains two motifs positioned independently</td>
</tr>
</tbody>
</table>
Transcription factor (TF) binding in regulatory elements

histone marks

nucleosomes

sequence motifs

Adapted from Shlyueva et al. (2014) Nature Reviews Genetics.
Cooperative vs non-cooperative binding

Defining Simulation Parameters

In [3]:
```python
print_simulation_info("simulate_heterodimer_grammar")
```

```
Parameters

motif1 : str, encode motif name
motif2 : str, encode motif name
seq_length : int, length of sequence
min_spacing : int, minimum inter motif spacing
max_spacing : int, maximum inter motif spacing
num_pos : int, number of positive class sequences
num_neg : int, number of negative class sequences
GC_fraction : float, GC fraction in background sequence
```

In [4]:
```python
heterodimer_grammar_simulation_parameters = {
    "motif1": "SPI1_known4",
    "motif2": "IRF_known20",
    "seq_length": 500,
    "min_spacing": 2,
    "max_spacing": 5,
    "num_pos": 10000,
    "num_neg": 10000,
    "GC_fraction": 0.4
}
```
Defining Simulation Parameters

```python
In [3]: print_simulation_info("simulate_heterodimer_grammar")
```
```
Parameters
---------
motif1 : str, encode motif name
motif2 : str, encode motif name
**seq_length** : int, length of sequence
min_spacing : int, minimum inter motif spacing
max_spacing : int, maximum inter motif spacing
num_pos : int, number of positive class sequences
num_neg : int, number of negative class sequences
GC_fraction : float, GC fraction in background sequence
```

```python
In [4]: heterodimer_grammar_simulation_parameters = {
    "motif1": "SPI1_known4",
    "motif2": "IRF_known20",
    "seq_length": 500,
    "min_spacing": 2,
    "max_spacing": 5,
    "num_pos": 10000,
    "num_neg": 10000,
    "GC_fraction": 0.4
}
```
Defining Simulation Parameters

In [3]:

```python
print_simulation_info("simulate_heterodimer_grammar")
```

```
Parameters
--------
motif1 : str, encode motif name
motif2 : str, encode motif name
seq_length : int, length of sequence
min_spacing : int, minimum inter motif spacing
max_spacing : int, maximum inter motif spacing
num_pos : int, number of positive class sequences
num_neg : int, number of negative class sequences
GC_fraction : float, GC fraction in background sequence
```

In [4]:

```python
heterodimer_grammar_simulation_parameters = {
    "motif1": "SPI1_known4",
    "motif2": "IRF_known20",
    "seq_length": 500,
    "min_spacing": 2,
    "max_spacing": 5,
    "num_pos": 10000,
    "num_neg": 10000,
    "GC_fraction": 0.4
}
```
Getting Simulation Data

Simulation name

Simulation parameters

Underlying Sequence:

```
“A G T A A A G A T T T T”
```
Convolutional Neural Networks

**Fully connected classification layer**: computes a probability using a logistic regression on the max pooling layer outputs.

**Max pooling layer**: Partition convolutional filter outputs and keep maximum output in each partition.

**ReLU transformation**: set negative motif scans to 0.

Scan sequence using with convolutional filters.

**Convolutional layer**: a set of convolutional filters similar to position specific scoring matrices (PSSMs).
Defining SequenceDNN models

cnn_parameters = {
    'seq_length': 500,
    'num_filters': 15,
    'conv_width': 25,
    'pool_width': 35
}
Defining Sequence DNN models

cnn_parameters = {
    'seq_length': 500,
    'num_filters': 15,
    'conv_width': 25,
    'pool_width': 35
}
Defining SequenceDNN models

```python
cnn_parameters = {
    'seq_length': 500,
    'num_filters': 15,
    'conv_width': 25,
    'pool_width': 35
}
```
Defining SequenceDNN models

cnn_parameters = {
    'seq_length': 500,
    'num_filters': 15,
    'conv_width': 25,
    'pool_width': 35
}
Training a DNN model

Training: Iterative updates to model parameters to minimize the “loss function”
Training a DNN model

The loss: quantifies error in model predictions

Predicted probability vs real class label
When to stop training?

```python
In [11]: SequenceDNN_learning_curve(cnn_model)
```
Performance Metrics

Epoch 0: validation loss: 0.699
Balanced Accuracy: 52.92%  auROC: 0.544  auPRC: 0.557  auPRG: 0.074
Recall at 5%|10%|20% FDR: 0.3%|0.3%|0.5%  Num Positives: 1645  Num Negatives: 1555

Epoch 73: validation loss: 0.134
Balanced Accuracy: 95.86%  auROC: 0.986  auPRC: 0.988  auPRG: 0.983
Recall at 5%|10%|20% FDR: 95.7%|97.0%|98.4%  Num Positives: 1645  Num Negatives: 1555

In [12]: test_SequenceDNN(cnn_model, simulation_data)

Test performance:
Balanced Accuracy: 95.88%  auROC: 0.987  auPRC: 0.988  auPRG: 0.984
Recall at 5%|10%|20% FDR: 95.8%|97.3%|98.5%  Num Positives: 2017  Num Negatives: 1983

Taken from Flach, Peter and Kull, Meeli. NIPS (2015).
Interpreting DNN models: two broad approaches

1. Model-centered approach: interpret model parameters directly
   - Example: inspect learned convolutional filters and try to infer sequence motifs from them

2. Input sequence-centered approach: sequence-specific model activity
   - Example: propagate input sequence through the model, inspect outputs in convolutional and max pooling layer, try to infer sequence properties from those output
Model-centered Interpretation

SPI1

IRF

Filter 1
Filter 2
Filter 3
Filter 4
Filter 5
Filter 6
Filter 7
Filter 8
Filter 9
Filter 10
Filter 11
Filter 12
Filter 13
Filter 14
Filter 15
Input sequence centered

Interpretation

15 Convolutional Filters, post max pooling

15 Convolutional Filters, post ReLU

15 Convolutional Filters, pre ReLU

Motif sites
WE HAVE TO GO
DEEPER!

DIYLOL.COM
Repeat with a deeper 3-layered CNN

```python
In [14]:
depth_sequenceDNN_parameters = {
    'seq_length': 500,
    'use_deep_CNN': True, # we have to specify this option when using a deep CNN
    'num_filters': 15,
    'conv_width': 15, # we decrease width of convolutional filters in the 1st layer
    'num_filters_2': 15, # define number and width of convolutional filters in 2nd and 3rd layers
    'conv_width_2': 15,
    'num_filters_3': 15,
    'conv_width_3': 15,
    'pool_width': 35,
    'verbose': 0} # we set verbose to 0 to suppress printouts during training

depth_cnn = get_SequencelDNN(depth_sequenceDNN_parameters)

In [15]:
train_SequenceDNN(depth_cnn, simulation_data)
SequenceDNN_learning_curve(depth_cnn)
test_SequenceDNN(depth_cnn, simulation_data)
interpret_SequenceDNN_distributed(depth_cnn, simulation_data)
```
Faster and Better Learning

Shallow CNN

Deep CNN
### Better Test Performance Metrics

#### Shallow CNN

<table>
<thead>
<tr>
<th>Test performance:</th>
<th>Balanced Accuracy: 95.88%</th>
<th>auROC: 0.987</th>
<th>auPRC: 0.988</th>
<th>auPRG: 0.984</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall at 5%</td>
<td>10%</td>
<td>20% FDR: 95.8%</td>
<td>97.3%</td>
<td>98.5%</td>
</tr>
</tbody>
</table>

#### Deep CNN

<table>
<thead>
<tr>
<th>Test performance:</th>
<th>Balanced Accuracy: 99.17%</th>
<th>auROC: 0.999</th>
<th>auPRC: 0.999</th>
<th>auPRG: 0.999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall at 5%</td>
<td>10%</td>
<td>20% FDR: 100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Plotting simulation motifs...
Model-centered Interpretation

Deep CNN

Filter 1
Filter 2
Filter 3
Filter 4
Filter 5
Filter 6
Filter 7
Filter 8
Filter 9
Filter 10
Filter 11
Filter 12
Filter 13
Filter 14
Filter 15
Motif sites

15 Convolutional Filters, post max pooling

15 Convolutional Filters, post ReLU

15 Convolutional Filters, pre ReLU

Shallow CNN

Deep CNN
THAT'S NOT ENOUGH
WE HAVE TO GO DEEPER
Integrative Interpretation of DNN Models

- Interpretation through internal layers DNN layers, both model-centric and input sequence-centric, suffers from the distributed nature of DNNs

- Solution: “integrate” using DeepLIFT and in-silico mutagenesis (ISM)
  - DeepLIFT: score each nucleotide based on its net contribution to the final fully connected layer, integrating across all filters and layers in between
  - ISM: mutate one nucleotide at a time, compute difference in prediction, score based on average difference in prediction
Integrative Input sequence scores
DeepLIFT can recover complex properties from DNN models!
Investigate the one layered CNN model used here for the following simulations:

1. single motif detection simulation of TAL1 in 1000bp sequence with 40% GC content
2. motif density localization simulation of 2-4 TAL1 motif instances in the central of 150bp of a total 1000bp sequence with 40% GC

Key questions:

1) What could explain the difference in ISM's sensitivity to the TAL1 motif sequence between the simulations?
2) What does that tell us about the the scope of ISM for feature discovery? Under what conditions is it likely to show sensitivity to sequence features?

Starter code is provided in the tutorial notebook.
To access this tutorial on Amazon AWS

1. Create an account on Amazon Web Services: [www.aws.amazon.com/signin](http://www.aws.amazon.com/signin)
2. You will need to launch an EC2 instance using the public AMI “DragonnTutorialPublic”
3. Go to Services > EC2 > AMIs
4. Select “Public Images”
5. In the search bar, enter “DragonnTutorialPublic”
6. Click “Launch” and follow the instructions. Note: you must select instance type “g2.2xlarge” or “g2.8xlarge” to create an instance with GPU’s
Acknowledgements

Deep Learners at Kundajelab:

Avanti Shrikumar
Nathan Boley
Peyton Greenside
Chris Probert
Nasa Armstrong
Irene Kaplow
Michael Wainberg
Oana Ursu
Rahul Mohan

Anna Shcherbina  Chuan Sheng Foo  Anshul Kundaje