Convolutional neural networks for classification of alignments of non-coding RNA sequences 复现

一、 摘要:

卷积神经网络（CNN）已被应用于 DNA 序列的分类问题，以四种核苷酸的离散化表示序列作为属的输入的 CNN 训练模型已成功学习到了权重矩阵，可以用用于预测蛋白质的结合位点。但是现有的以单一序列作为输入的 CNN 训练模型还存在着不足，例如不能解决序列上的增删问题。原作者提出了一种将成对比对序列与 RNA 二级结构结合作为输入的 CNN 训练模型，实现了精准的序列聚类效果；此外，对 ncRNA 族的聚类与预测效果也优于现有方法。

本文是基于 Convolutional neural networks for classification of alignments of non-coding RNA sequences 复现过程的介绍，主要介绍了数据获取、原文整体思路、核心代码分析、结果展示、任务分工介绍。

二、 数据获取:

作者已于网站 http://www.dna.bio.keio.ac.jp/cnn/分享了该实验的相关数据，但我们还是根据文章中说介绍的方法下载了有关数据。

原文数据集源自于 Ensembl，Ensembl 提供的 ncRNA 的序列数据是对多个数据库（例如 Rfam 和 HUGO 基因命名委员会（HGNC）数据库（RNA 家族数据库）中存在的 ncRNA 的序列数据的整合。由于 Ensembl 提供的数据中不包含 tRNA 的序列数据，因此原文选择从基因组 tRNA 数据库（GtRNAdb）中检索。作者在性能分析中测试了 9 个 ncRNA 家族，数据集中每个家族的基因数量如表 1 所示。

snRNA	snoRNA C/D	snoRNA H/ACA	scaRNA	miRNA	YRNA	Vault RNA	5S rRNA	tRNA
2053	322	160	56	1890	831	11	29	631

表 1

在下载数据的过程中，我们发现组内三人都没有办法进入 Ensembl 数据库的下载界面。于是我们尝试转向 HGNC 数据库下载 ncRNA 数据，但是依旧无法进入 HGNC 的下载界面。经过反复尝试，我们从 GtRNAdb 数据库下载了 tRNA 的序列数据，如图 1 所示。

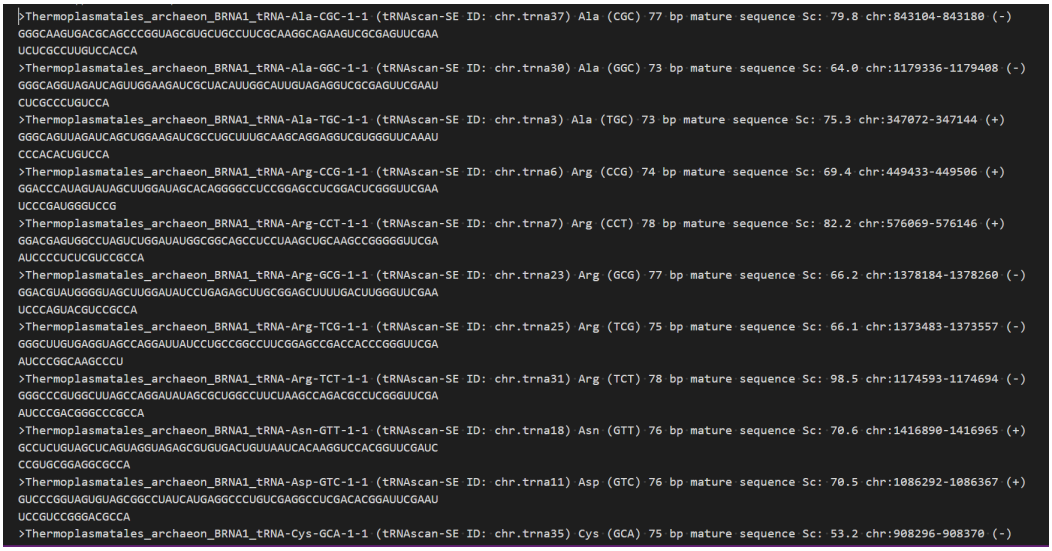

图 1

三、 整体思路:

1. 核苷酸、二级结构的离散化表示：

输入到一维 CNN 中两条序列需要以离散形式进行表示。代表四种核糖核苷酸的 A、C、G、U 以及表示空缺的“-”可以用一组五维独热编码的向量表示出来。此外， ncRNA 特有的二级结构信息可以由一个三维向量来表示。ncRNA 折叠成功能分子的过程由 A–U 和 C–G 的形成决定，这些碱基对构成 ncRNA 的二级结构。RNA 序列中第 i 和第 j 个核苷酸形成碱基对的碱基配对概率p_{ij}可以通过 McCaskill 算法计算。接下来对于每个位置 i，我们将碱基配对概率分为三种：与下游核苷酸配对的左侧配对概率$p_i^{left} = \sum_{j>i} p_{ij}$；与上游核苷酸配对的右侧配对概率$p_i^{right} = \sum_{j<i} p_{ij}$；未配对概率$p_i^{unpaired} = 1 - (p_i^{left} + p_i^{right})$。因此，用于表示第 i 列二级结构信息的三维向量由左侧碱基配对概率p_i^{left}，右侧碱基配对概率p_i^{right}和未配对概率$p_i^{unpaired}$构成，如图 2 所示。

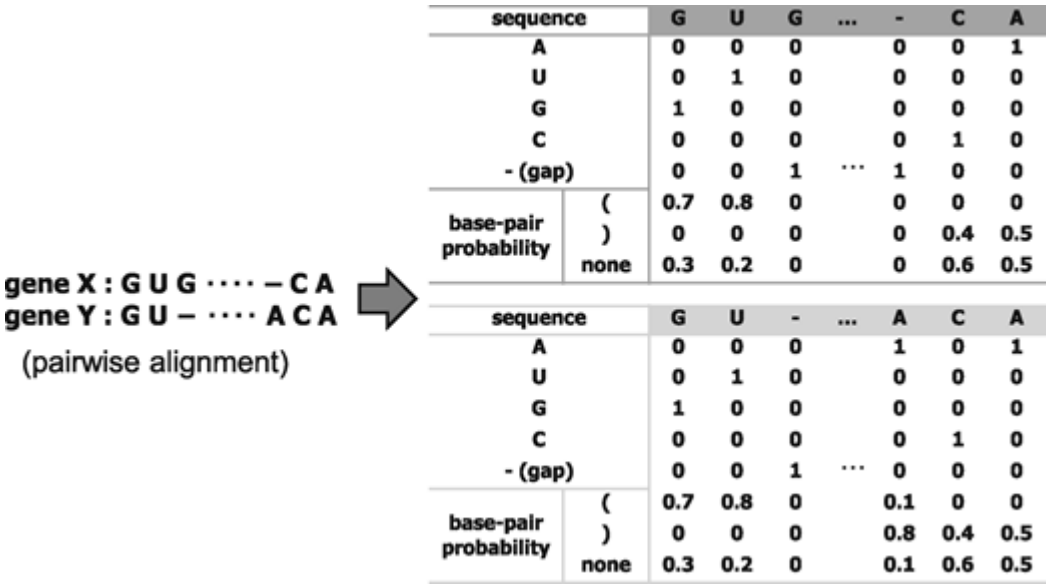

图 2

其他的离散性表示 RNA 或 DNA 序列的方式是应用 word2vec 将子序列转化为 n 维向量。对于 RNA 或 DNA 序列，每个长度为 k 的子序列都被视为一个单词，并通过 word2vec 转换为向量。

2. 构建 CNN：

在本实验中的 CNN 模型包含两组卷积层+池化层结构，其后是三层全连接层，如图 3 所示。

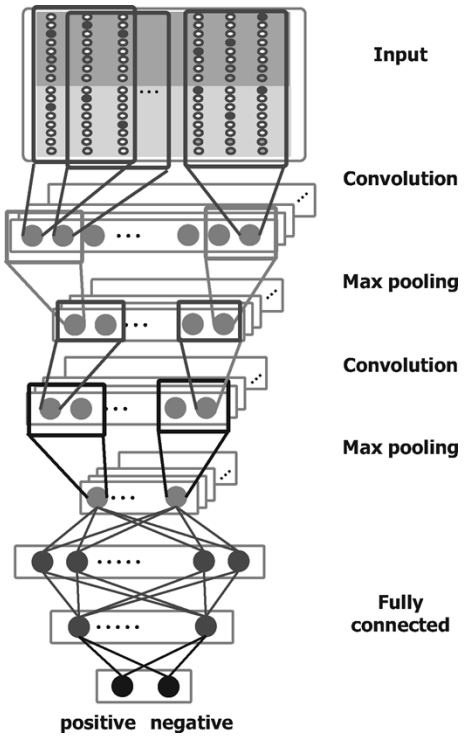

图 3

在学习 CNN 期间，调整了六个超参数（卷积内核大小、内核数目、池化大小、隐藏层中的单元数目和学习算法）。表 2 列出了每个超参数的调整范围。使用大小为 128 的小样本数进行样本归一化，并使用 Dropout 防止过度拟合。

Hyperparameter	**Range**
Kernel size for convolution	3, 7, 15, 20, 30, 40, 50
Number of kernels (in two convolution layers)	6:13, 13:26, 19:38, 32:64, 45:90, 64:128, 128:256
Pooling method	Max pooling, average pooling,
Pooling in second layer	Global max pooling, local max pooling
Number of units in hidden layer (ratio to input layer)	1/3, 1/2, 2/3, 3/4, 1
Learning algorithm	Adam, AdaGrad, AdaDelta, Momentum SGD

表 2

3. 对预测结果聚类：

经过 CNN 预测后会生成一个相似性得分矩阵，可将及其转化为带有‘0’、‘1’标签的分类矩阵，该矩阵可以视为图表示的邻接矩阵。理论上讲，从邻接矩阵表示的整个图中提取的每个完整的最大子图就对应于包含相似 ncRNA 序列的聚类。在本文中，作者选择 k-means 来对分类矩阵进行聚类，如图 4。

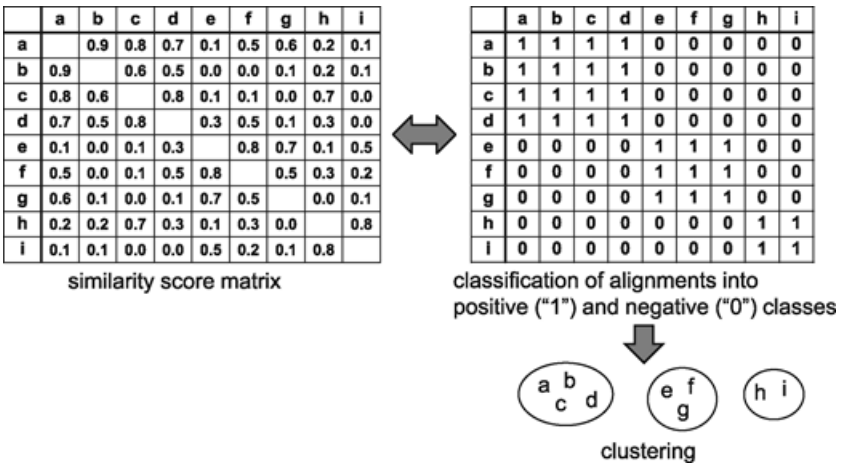

图 4

4. 与单序列 CNN 的对比：

为了证明将成对序列作为输入的 CNN 模型可以产生更优的效果，作者将以单一序列作为输入的 CNN 模型与将成对序列作为输入的 CNN 模型进行比较。在比较中，用离散型单序列作为一维 CNN 的输入，如果序列属于目标家族，则训练数据中每个序列的标签为正，否则为负。在对多个家族的 ncRNA 序列进行聚类的实验中，使用一维 CNN 进行多标签学习，一维 CNN 的输出节点数与 ncRNA 家族的数目相同。

四、 核心代码分析:

1. 序列的离散化表示：

代表四种核糖核苷酸的 A、C、G、U 以及表示空缺的“-”可以用一组五维独热编码的向量表示，如图 5 所示。

```
for k in range(len(pair1)):
    ## pair1's data
     # bases data
    if pair1[k] == "A":
        data[k][0] = 1
    elif pair1[k] == "T":
        data[k][1] = 1
    elif pair1[k] == "G":
        data[k][2] = 1
    elif pair1[k] == "C":
        data[k][3] = 1
    elif pair1[k] == "-":
        data[k][4] = 1
     # base-pair probability data
    if pair1[k] != "-":
        data[k][5] = bp_mat[0][n1][0]
        data[k][6] = bp_mat[0][n1][1]
        data[k][7] = bp_mat[0][n1][2]
        n1 += 1
```

图 5

RNA 序列的二级结构信息可以由一组三维向量来表示：与下游核苷酸配对的左侧配对概率$p_i^{left} = \sum_{j>i} p_{ij}$；与上游核苷酸配对的右侧配对概率$p_i^{right} = \sum_{j<i} p_{ij}$；未配对概率$p_i^{unpaired} = 1 - (p_i^{left} + p_i^{right})$，如图 6 所示。

```
## Calcurate base-pair probability
for i in range(len(seqlen)):
    bp = np.zeros([seqlen[i], 3], dtype=np.float32)
    # each base
    for j in range(seqlen[i]):
        Left = 0
        Right = 0
        # probability of (
        for k in range(j, seqlen[i]):
            Left += Array[i][j][k]
        # probability of )
        for k in range(j):
            Right += Array[i][k][j]
        bp[j][0] = Left
        bp[j][1] = Right
        bp[j][2] = 1 - Left - Right

    bp_mat = bp_mat + [bp]
#print(bp_mat)
```

图 6

经离散化表示后的成对比对序列矩阵由图 7 所示：

```
 [[0.        0.        1.        ... 0.83992553 0.        0.1600745 ]
  [0.        0.        0.        ... 0.98162788 0.        0.0183721 ]
  [0.        1.        0.        ... 0.98711503 0.        0.012885  ]
  ...
  [0.        0.        0.        ... 0.         0.        0.        ]
  [0.        0.        0.        ... 0.         0.        0.        ]
  [0.        0.        0.        ... 0.         0.        0.        ]]

 [[0.        0.        1.        ... 0.89991498 0.        0.100085  ]
  [0.        0.        0.        ... 0.98955297 0.        0.010447  ]
  [0.        1.        0.        ... 0.98537999 0.        0.01462   ]
  ...
  [0.        0.        0.        ... 0.         0.        0.        ]
  [0.        0.        0.        ... 0.         0.        0.        ]
  [0.        0.        0.        ... 0.         0.        0.        ]]

 [[0.        0.        1.        ... 0.89991498 0.        0.100085  ]
  [0.        0.        1.        ... 0.98955297 0.        0.010447  ]
  [0.        0.        0.        ... 0.98537999 0.        0.01462   ]
  ...
  [0.        0.        0.        ... 0.         0.        0.        ]
  [0.        0.        0.        ... 0.         0.        0.        ]
  [0.        0.        0.        ... 0.         0.        0.        ]]]
(1770, 1200, 16)
```

图 7

2. CNN 模型构建：

原文应用 chainer 框架构建了一个包含两组卷积层+池化层结构紧跟三层全连接层的 CNN 模型，并使用 Dropout 防止过度拟合，如图 8 所示。chainer 是一个专门为高效研究和开发深度学习算法而设计的开源框架，可以在在训练时“实时”构建计算图。

```
class CNN(Chain):
    def __init__(self, output_ch1,output_ch2, filter_height, filter_width, n_units, n_label):
        super(CNN, self).__init__(
            conv1 = L.Convolution2D(1, output_ch1, (filter_height, filter_width)),
            bn1 = L.BatchNormalization(output_ch1),
            conv2 = L.Convolution2D(output_ch1, output_ch2, (filter_height, 1)),
            bn2 = L.BatchNormalization(output_ch2),
            fc1 = L.Linear(None, n_units),
            fc2 = L.Linear(None, n_label))

    def __call__(self, x, train=True):
        h1 = F.max_pooling_2d(F.relu(self.bn1(self.conv1(x))), ksize=(10,1), stride=8)
        h2 = F.max_pooling_2d(F.relu(self.bn2(self.conv2(h1))), ksize=(14,1), stride=8)
        h3 = F.dropout(F.relu(self.fc1(h2)), ratio=0.5)
        y = self.fc2(h3)
        return y
```

图 8

根据原文来看，作者应用了由 CUDA 加速的 GPU 集群来训练模型。在运行这一步时，由于缺少相应的 GPU 并且无法安装 CUDA 加速驱动，我们没有运行出对应的结果。由于原文的 CNN 结构是由 chainer 框架构建的，我们三个人之前没有接触过这一框架。目前我们还没有找到绕过 CUDA 加速，直接应用 CPU 训练的方法。根据原文来看，训练后的结果应该由图 9 所示。

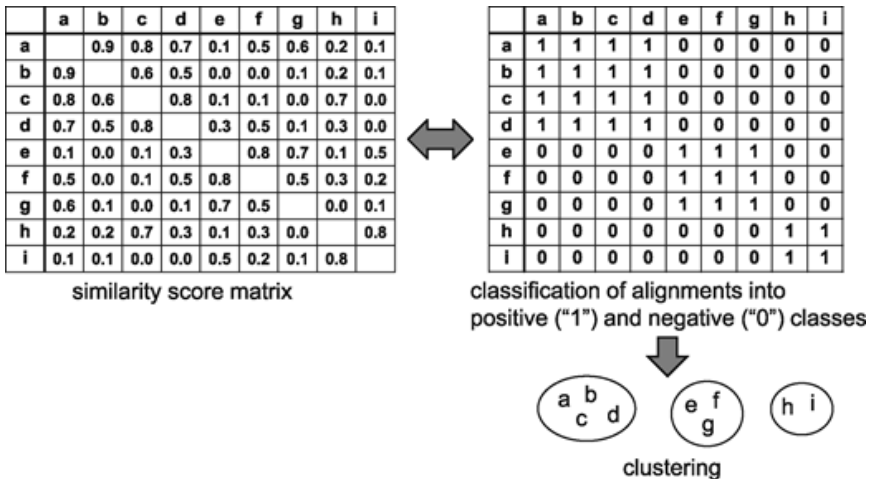

图 9

五、 结果展示:

1. 一维 CNN 分类精度：

Method	Accuracy	F-value
CNN with DAFS (word2vec)	**0.980**	**0.931**

Method	Accuracy	F-value
CNN with DAFS (one-hot coding)	0.971	0.901
CNN with DAFS (only secondary structure)	0.943	0.803
CNN with Clustal Omega (one-hot coding)	0.958	0.850

表 3 一维 CNN 分类精度

应用 DAFS 成对结构对齐方式和 word2vec 离散表示形式的 CNN 在准确度和 F 值方面都产生了几乎完美的预测。在两种不同的离散表示形式之间，word2vec 的性能优于独热编码。在两种类型的比对之间，DAFS 结构比对表现出比 Clustal-Omega 序列比对更好的性能。但另一方面，仅输入二级结构信息不足以获得准确的预测。

2. ncRNA 序列聚类效果：

Method	Accuracy	F-value
10-fold CV		
CNN with DAFS (word2vec)	**0.957**	**0.868**
CNN with DAFS (one-hot coding)	0.939	0.824
CNN with DAFS (only secondary structure)	0.910	0.731
CNN with Clustal Omega (one-hot coding)	0.927	0.784
RNAclust	0.890	0.580
Ensembleclust	0.887	0.654
Spectral clustering based on DAFS	0.855	0.554
Unknown family		
CNN with DAFS (word2vec, six families)	**0.752**	0.646

Method	Accuracy	F-value
CNN with DAFS (word2vec, three families)	0.717	0.586
CNN with DAFS (one-hot coding)	0.685	0.560
RNAclust	0.707	0.208
Ensembleclust	0.711	**0.650**
Spectral clustering based on DAFS	0.664	0.588

表 4 ncRNA 家族聚类准确性的效果对比

在 10 折交叉验证中，基于 CNN 的方法在准确性和 F 值方面均优于三种现有方法，特别是产生了比现有方法更好的 F 值。在未知家庭验证中，六个家族 CNN 的训练方法显示出最高的准确性，高于三个家族的 CNN 方法。 F 值在 Ensembleclust 和基于六个族的 CNN 方法之间具有可比性。

3. 与单个序列的输入比较：

Method	Accuracy	F-value
CNN with input of single sequence	0.913	0.734
CNN with DAFS alignment	**0.939**	**0.824**

表 5 与输入单个序列的一维 CNN 的效果比较

与输入单个序列的 CNN 相比，输入 DAFS 成对比对序列的 CNN 的准确性和 F 值方面的预测效果要好，从而证明了输入成对比对序列的优势和实用性。

4. 输入包含其他信息（转录组数据）时的效果：

Method	Accuracy	F-value
CNN with SHARAKU (one-hot coding)	**0.907**	**0.815**
CNN with DAFS (one-hot coding)	0.903	0.811
Spectral clustering based on SHARAKU	0.747	0.516

表 6 以读序列定位图谱作为附加信息时，对聚类准确性影响

与具有 DAFS 结构比对序列的 CNN 相比，输入带有序列、二级结构和读序列定位图谱的 SHARAKU 成对比对序列的 CNN 准确性略好。因此，输入额外的读序列定位图谱，有助于提高 ncRNA 家族的聚类效果。

六、 相关贡献:

在复现的过程中，邓景耀同学完成了相关数据的搜索、下载；麦湘健同学主要负责实验环境的搭建、分析与运行实验代码；王子烨同学负责整体思路的梳理、PPT 制作及文档撰写、实验成果汇报。