ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте

на тему: Программа обнаружения изменений на спутниковых данных

(промежуточный, этап 1)

Выполнил:
Студент группы БПМИ201 Аржанцев Андрей Иванович Аржанцев
Подпись И.О.Фамилия

13.02.2022
Дата

Принял:
Руководитель проекта
Имя, Отчество, Фамилия

Должность, ученое звание

Место работы (Компания или подразделение НИУ ВШЭ)

Дата проверки 2022
Оценка (по 10-ти бальной шкале) Подпись

Москва 2022

Содержание

Аннотация

Реализация алгоритма с использованием Fuzzy ARTMAP нейросети, находящего и классифицирующего изменения в двух изображениях одного места в разное время.

1 Введение

Моя команда делает проект по реализации приложения для определения изменений на географических картах с течением времени. Существует множество алгоритмов, решающих данную задачу, и наша общая задача написать приложение, в которое можно загрузить два изображения одного и того же места в разное время, выбрать один из реализованных нами алгоритмов и на выходе получить наглядную информацию об изменениях, произошедших на изображении.
Моя часть проекта заключается в том, чтобы реализовать один из алгоритмов, решающих данную задачу. Я выбрал алгоритм, основанный на fuzzy ARTMAP нейронной сети. Этот алгоритм решает более общую задачу - выделение областей на изображении принадлежащих одной природной зоне. Делает это он, как и любая нейросеть, вследствие предварительного обучения на какой-то выборке с известным ответом, после чего умеет с хорошей точностью находить ответ для любых входных данных. Понятно также, что решая более общую задачу, мы без труда справимся и с изначально поставленной, но как именно также будет описано в основной части.
Источники, по которым я изучал принцип работы алгоритма, перечислены в самом конце.

2 Описание требований к проекту

Реализовывать свой алгоритм я буду на Python. Так как моя реализация должна быть включена в общий проект, то моя задача также заключается в том, чтобы входные и выходные данные можно были использовать в самом нашем приложении.
Также для обучения своей нейронной сети и для последующей проверки работы алгоритма, мне нужен будет какой-то dataset изображений. В интернете я нашел множество таких, например [3] из литературы, который показался мне наиболее подходящим, ведь в нем как раз есть dataset'ы с изображениями с разметкой по природным зонам.

3 Основная часть

Постановка задачи
Пусть на вход программы поступает последовательность изображений (желательно крупномасштабных) одного и того же места, но в разное время.
На выходе мы хотим получать информацию о том, как изменялась среда в этом месте, например, где были построены какие-то объекты, вырублены леса, где пересохли реки и т.д.
Дополнительно пусть у нас есть набор изображений этого же или других мест вместе с тем же изображением в формате 'физической карты', то есть цветом выделены разные природные зоны (леса, горы, населенные пункты, водные объекты и т.д). Далее эти природные зоны я буду называть классами.

Основная идея
Попробуем решить сначала такую задачу: научиться строить по изображению его "физическую карту". То есть мы хотим уметь выделять на изображении области, принадлежащие одному и тому же классу.

Если мы научимся это делать, то сравнив значение классов в каждом пикселе двух изображений разного времени, мы сможем посмотреть, изменился он или нет, и, более того, мы сможем понять из какого класса в какой данный пиксель перешел.
Решать новую задачу я буду с помощью алгоритма с нейронной сетью, которая будет подробно описана позже. Здесь же хочется сказать о проблемах, которые у нас могут возникнуть при использовании нейросети для этой задачи.
Во-первых, если в нашей обучающей выборке не будет представлен какой-то класс, который будет на изображениях из входных данных, мы заведомо будем получать неточный ответ. Поэтому дополнительным условием задачи будет: уметь находить представителей класса, который не встречается в обучайющей выборке.
Во-вторых, решение, полученное нейросетью будет явно не идеальным, поэтому на самом деле помимо явного определения класса для данного пикселя, мы также хотим иметь набор вероятностей, что данный пиксель принадлежит тому или иному классу. Это даст нам больше информации о непосредственном изменении данного пикселя с течением времени, и, комбинируя явное значение класса и вероятностную принадлежность тому или иному классу, мы можем точнее сказать об изменениях.

Описание работы нейросети

Я буду пользоваться fuzzy ARTMAP нейронной сетью для нахождения класса по данному пикселю изображения. Основную идею я прочитал в статье [1] и добавил идею для реализации нахождения новых классов из статьи[2]
Как она выглядит: в ней три уровня вершин - I(входные данные - параметры нашего пикселя, мощность $2N$), F(представители "размытого"класса, мощность M) и C(выходные данные - непосредственно классы, мощность L).$M \geq L$, причем, чем больше M, тем точнее результат(фактически мы предполагаем, что классный класс на самом деле может делиться на несколько подклассов)
Также вершины разных уровней соединены взвешенными ребрами с весами $\{W_{ij} \in [0,1] | i \in I, j \in F\}$ и $\{W^{end}_{jk} \in \{0,1\} | j \in F, k \in C\}$.
Сначала покажем, как наша нейросеть будет обучаться на данном нам наборе.
Для этого нам потребуются еще 4 переменных: $\alpha > 0$(параметр выбора), $\beta \in [0,1]$(обучающий параметр), $p \in [0,1]$(парамтер бдительности), ϵ(параметр нахождения совпадений). Изначально пусть $\alpha = 0.001, \beta = 1, p = 0, \epsilon = 0.001$(значения взяты, как стандратные из интернета, в процессе реализации они, возможно, поменяются). Изначальные веса берем любые, между F и C из каждой вершинки нужно ровно одно ребро веса 1.
Пусть в какой-то момент обучения нам пришли входные данные $A = (a_1, ...a_n)$. Сделаем из них $A = (a_1, ..., a_n, 1 - a_1, ...1 - a_n)$.
Рассмотрим функцию выбора $T_J(A) = \frac{|W_j \wedge A|}{\epsilon + |W_j|}$, где $a \wedge b = \min(a, b), |a| = \sum a_k$.
Отсортируем все вершины второго уровня по возрастанию $T(J)$ и начнем их перебирать.
Для текущего J проверяем, что выбор удовлетворяет нашей текущей бдительности, то есть $\frac{|W_j \wedge A|}{2M} > p$. Если это неверно, переходим к следующей вершине. Если же верно, то проверяем корректность, то есть, что мы правильно определили класс для наших входных данных, $W^{end}_{Jk} = 1$ для правильного. Если не корректно, то увеличиваем p по формуле $p = \frac{|W_j \wedge A|}{2M} + \epsilon$ и опять-таки переходим к следующей вершине. Если корректно, то пересчитываем веса по формуле: $W^{new}_J = \beta(A \wedge W^{old}_J) + (1 - \beta)W^{old}_J$ и переходим к следующему входному вектору A.
Если же случилось так, что ни одна вершина не прошла требуемые два условия, мы меняем соответствие между F и C, делая переход из нашего первого выбранного J в правильный $K \in C$ с весом 1, другие делаем весами 0.
Обучившись, нейросеть теперь может определять принадлежность классу для любого входа - для этого надо посмотреть на соответствуйщий ей J(лучшая при сортировки по $T(A)$) и посмотреть в какой класс переходит J (уже, очевидно, никаких двух проверок мы уже делать не можем).

Теперь о том, как я буду определять изменения для двух изображений. Во-первых, дадим нашей нейросети на вход набор пикселей сначала из "старого"изображения, потом из "нового". Наша модель умеет определять для каждого пикселя класс, которому он принадлежит. Но также для каждого пикселя у нас есть набор переходов в "размытые"классы, то есть значение W обратно пропорциональное функции выбора T, нам понадобится это для двух целей.
Первая - уметь определять, что пиксель на самом деле не относится ни к одному из существующих классов из обучения. Делать это будем так: зафиксируем какое-то $(CD) > 1$ и будем проверять выполнения условия $W(J) > (CD)W(I)$, где J - класс, который выдает как ответ наша нейросеть, I - следующий по весу класс, которому может принадлежать пиксель. Грубя говоря, если мы попали в ситуацию, когда определенный класс

не мажорирует над другими, очень вероятно, что мы попали в ситуацию, когда это какой-то другой класс, признаки которого схожи с какими-то разными классами из обучения. Таким образом, мы даже имеем представление о том, чем может являться этот класс, посмотрев на то, на какие из обучаемых классов он похож. Второе - мы можем получить приблизительную информацию о точности определения класса для данного пикселя. Для этого просто посмотрим на отношение веса для определенного нами класса и суммы весов по всем классам. Сделав так для старого и нового изображения для двух соответствующих пикселей и перемножив значения, получим приблизительную вероятность ошибки. То есть помимо полученной карты изменений, мы будем также иметь точность определения изменения класса для каждого пикселя.

Скажу еще немного о том, как можно оптимизировать скорость и точность обучения нашей нейросеть, и как долго ей надо обучаться.
Во-первых, можно разделить весь данный набор изображений на несколько (10) равных частей, после чего проводить обучения на каждом из них. Делать обучения можно попеременно на каждом блоке, каждый раз обновляя веса и переходя к другому блоку. При этом в одному и тому же блоку нам необходимо периодически возвращаться, ведь точность обучения нашей сети зависит от порядка, в котором мы даем ей входные данные.
Во-вторых, мы можем разделить входные параметры для каждого пикселя также на несколько групп. То есть, сначала можно обучаться на небольшом количестве параметров из каждой группы, а потом объединять группы, передавая нейросети уже полученные веса для каждой текущей посчитанной группы параметров и пересчитывая их. Это улучшит и точность и скорость вычислений, так как будут лучше и быстрее посчитаны ответы для зависимости отдельных групп параметров. Например, если какие-то параметры будут ключевыми, то их значимость мы сможем вычислить еще на этапе вычислений с небольшим числом параметров.

Параметры входных данных
Теперь о том, в каком виде мы будем передавать нашей сети входные данные. Пусть нам дано изображение из пикселей, мы хотим иметь количесвенные значения каких-то признаков по всем пикслям.
Во-первых, мы можем передать в качестве параметров значение самого пикселя, его географические координаты. Также передадим количественные параметры каждого пикселя, которые считаются по известным формулам ([4]), такие как NIR(изображение при инфрокрасном излучении), $NDVI$(индекс растительности), $NDWI$(индекс воды), $NDSI$(индекс снежного покрова), $NDBI$(индекс застройки) и, возможно, другие.
Эти индексы во многом невилируют проблему из-за разной освещенности и погодных условий на разных изображениях. Тем не менее, если мы имеем возможность узнать время и погоду на изображении, то мы можем добавить это значение в качестве параметра у соотвветствующих пикселей.
Также, во избежании проблем с неудачными отдельно взятыми пикселями, мы можем разбить изображение на небольшие квадраты пикселей, либо для каждого пикселя рассматривать также ближайшие к нему пиксели. Для группы мы также можем найти все нужные параметры, после чего можно выкинуть значения у пикселей, в которых сильно отличаются какой-то параметр, а остальные параметры усреднить. Таким образом, мы навряд ли потеряем в точности, ведь группа близко лежащих пикселей скорее всего почти целиком принадлежит одному и тому же классу, но зато избавимся от проблемы с "плохими"пикселями.
Наконец, получив значения параметров, с помощью линейной функции сделаем их лежащими в отрезке $[0, 1]$. Это необходимо для того, чтобы было соответствие между весами ребер в нейросети, где они также лежат в отрезке $[0, 1]$.

Заключение
Представленный алгоритм действительно решает все поставленные задачи и даже дает более детальную информацию о входных изображений. Ответом же на непосредственно нашу задачу будет, во первых, матрица размера входного изображения, где на месте каждого пикселя стоит пара (значение класса пикселя на старом изображении; значение класса пикселя на новом изображении). Для большей наглядности можно каждой паре задать в соответствие какой-то цвет и выводить цветное изображение. Во-вторых, у нас также будет матрица, в которой для каждого пикселя показана точность определения смены класса, которую также можно представить и выводить в виде цветного изображения.

4 Литература

1. Статья с описанием работы Fuzzy ARTMAP нейронной сети
https://clck.ru/bDamQ

2. Статья о применении Fuzzy ARTMAP нейронной сети
https://www.researchgate.net/publication/248480978_Change_Detection_Using_Adaptive_Fuzzy_Neural_Networks
3. Datasets
https://earthexplorer.usgs.gov/
4. Галерея иидексов
https://pro.arcgis.com/ru/pro-app/latest/help/data/imagery/indices-gallery.htm
Другие использованные статьи и сайты:
5.https://clck.ru/bDU7p
6.https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9136674
7.https://en.wikipedia.org/wiki/Adaptive_resonance_theory
8.https://clck.ru/bDYL5