Обучение представлений и глубокое обучение, Домашнее задание №1,2

Kashin Andrey

11 мая 2015 г.

Весь код доступен по ссылке `https://github.com/IIoTeP9HuY/dl-course/tree/master/Homework`

1 Часть 1

Качество на первом задании:

- Public: 0.86765
- Private: 0.86945

При решении домашнего задания я пользовался библиотекой caffe. Также, я воспользовался этой модификацией чтобы иметь возможность использовать бОльшие батчи, при той же памяти видеокарты: `https://github.com/shelhamer/caffe/tree/accum-grad` Она позволяет агрегировать градиент по батчам и применять его после просмотра заданного количества батчей.

1 Архитектуры

Я пробовал несколько архитектур:

- Архитектуры из 5-го семинара
- Архитектуры из примера `https://github.com/BVLC/caffe/blob/master/examples/hdf5_classification.ipynb`
- LeNet
- Модифицированный LeNet, по одному новому слою каждого типа и DropOut
- CaffeNet
- Multi-Column Deep Neural Networks

Из них мне удалось обучить только первые четыре. Лучший результат дала последняя архитектура которая представляла собой Lenet с добавлением еще одного сверточного слоя после сверточных и еще одного полносвязного перед полносвязными. В самом конце был добавлен DropOut.

Для последних двух моделей качество на обучающей и тестовой выборке не уменьшалось, и logloss колебался в приделах 7 (на правильно обученой сети он был < 1).

2 Предобработка

Картинки я отмасштабировал до размера 50×50. Нормировал их интенсивности, вычитал среднее. Без нормировки и вычитания, Lenet учиться отказывался. Так как я использовал формат hdf5, автоматической нормировки не происходило. Я выбрал такой размер, так как с ним я все еще мог различать иероглифы одного и того же класса.

Еще я пытался применить фильтр dilate из opencv для преобразования картинок и обогащения обучающей выборки. Это немного улучшило качество. Я верю, что если немножко повращать и поискривлять картинки и добавить их в обучающую выборку, это может сильно улучшить результат.

При работе с данными, я разбил обучающую выборку на две подвыборки, train и test в пропорциях 90% и 10%. При этом разбиение было равномерным по классам, т.е. в train было 90 объектов каждого класса, в test было 10 объектов каждого класса.

3 Трюки

Я пытался воспользоватся методами semi-supervised learning. Предсказать метки на тестовой выборке, взять объекты, в которых мы очень уверены и добавить их в обучающую выборку. Этот подход не сильно помог.

Пытался воспользоваться методом Fine-Tuning взяв веса с сети обученной на ImageNet, но эта архитектура не захотела обучаться.

Экспериментировал с дообучением сети путем наращивания количества классов, на которых мы обучаемся. Брал первых 100 классов, обучал сеть на них. LogLoss уходил в 0. Добавлял классов до 500, обучался. Когда классов становилась 1000, LogLoss уже не опускался до 0. Результирующая сеть для 2000 была сопоставимой с сетью, которая изначально обучалась на 2000. При этом сложные архитектуры не хотели обучаться даже на малом количестве классов.

4 Анализ ошибок

Я смотрел на иероглифы, на которых сеть меньше всего уверена. Их метка действительно часто была ошибочной. При этом сами примеры иероглифов были очень сложными, с большим количеством деталей. И ошибочный класс отличался от реального наличием или отсутствием какой-то мелочи типа штриха.

2 Часть 2

Качество на втором задании:

- Public: 0.92352
- Private: 0.92570

1 Подходы

Сначала я воспользовался решением первого задания, предсказывал для иероглифа метку и сравнивал метки. Качество было достаточно низким ≈ 0.63. Первая попытка с сиамской сетью дала качество ≈ 0.7.

2 Архитектуры

Использовал стандартный сиамский LeNet. Пытался использовать сеть из первого задания, но она была слишком сложной и никакого прироста по качеству не давала, а работала очень долго. При этом даже если взять веса с задачи классификации, качество не удается увеличить.

3 Данные

Для обучения я брал данные из первого домашнего задания и генерировал пары иероглифов для классификации в пропорциях 50% / 50% по положительным и отрицательным парам.

В этом соревновании добавление новых пар в обучающую выборку помогало очень сильно, но, когда в обучающей выборке стало 500000 пар, качество на тестовой выборке расти перестало.

4 Трюки

Довольно важным трюком был выбор расстояния, после которого нужно считать иероглифы различными. Для этого я просто подбирал его таким образом, чтобы количество различных и одинаковых пар на тестовой выборке было одинаковым. Это простая процедура, которую можно осуществить бинарным поиском.