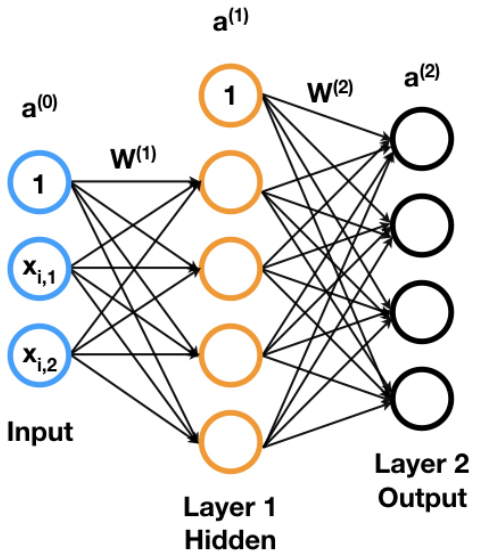
Assignment 3. Sentiment analysis using feed-forward neural networks

14 ноября 2020 г.

Теоретическая часть

Введем следующие обозначения:
$(x_{\{1\}}, y_{\{1\}}), \ldots, (x_{\{N\}}, y_{\{N\}})$ — обучающая выборка размера N, $x_{\{i\}} \in \mathbb{R}^{1\times M}$ — i-й отзыв из выборки, $M = s^{(0)}$ — размерность входных векторов, $y_{\{i\}} \in \{0, \ldots, K-1\}$, $s^{(l)}$ - количество нейронов в l-м слое, $W^{(l)}$ — матрица параметров l-го слоя размера $(s^{(l-1)}+1) \times s^{(l)}$ (т.к. мы добавляем смещение — bias), где $l = \{1, 2, \cdots, L\}$, L - количество слоев (число скрытых слоев равно L-1).

Пример полносвязной нейросети с двумя слоями (одним скрытым слоем):



Forward pass (for i-th example):

$$a^{(0)} = x_{\{i\}}$$
$$z^{(1)} = [1, a^{(0)}]W^{(1)}$$
$$a^{(1)} = tanh(z^{(1)})$$
$$z^{(2)} = [1, a^{(1)}]W^{(2)}$$
$$\hat{y}_{\{i\}} = a^{(2)} = softmax(z^{(2)})$$

Backward pass:

...

1. Посчитайте производную функции $\tanh(z)$ и выразите ее через саму функцию $tanh(z)$, считая что z — скаляр. Преобразуйте ответ, так, чтобы при вычислении $\tanh(z)$ и ее производной была только одна операция экспоненцирования.

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

2. Воспользовавшись обозначениями, введенными выше, выпишите формулы прямого прохода (forward pass) и вычисления оценочной функции кросс-энтропия

$$ce(W^{(1)}, \ldots, W^{(L)}, x, y)$$

для одного примера для полносвязной нейронной сети с L–1 скрытым слоем для случая многоклассовой классификации (считаем, что есть K взаимоисключающих классов). В качестве активации для скрытого слоя используется $tanh(z)$, для выходного слоя — $softmax(z)$.

3. Выпишите формулы прямого прохода и вычисления оценочной функции

$$CE(W^{(1)}, \ldots, W^{(L)}, x_{\{1\}}, \ldots, x_{\{N\}}, y_{\{1\}}, \ldots, y_{\{N\}})$$

для батча из N примеров в векторизованном виде (без цикла по примерам). Рядом с каждой формулой укажите размеры всех матриц. Считайте, что батч представлен матрицей $X \in R^{N \times M}$ и матрицей one-hot векторов правильных ответов $Y \in R^{N \times K}$.

4. Покажите, что $softmax(z + c) = softmax(z)$, где c – вектор, все компоненты которого равны. Как можно воспользоваться этим свойством при реализации $softmax$, чтобы не экспоненцировать большие положительные числа (что может привести к переполнению числа с плавающей точкой)?

5. Посчитайте, сколько всего параметров содержится в полносвязной нейронной сети с L–1 скрытым слоем, если входные вектора имеют размерность M, выходные вектора — K, а в каждом скрытом слое — H нейронов.

6. Для случая одного входного примера выведите формулу для $\delta^{(L)}$ — градиента оценочной функции по предактивациям в последнем слое $z^{(L)}$. Сначала выведите формулу для одной компоненты, затем для всего вектора.

7. Для случая одного входного примера выведите формулу для подсчета $\delta^{(l)}$ — градиента оценочной функции по $z^{(l)}$ — через $\delta^{(l+1)}$. Сначала выведите формулу для одной компоненты, затем для всего вектора.

8. Для случая одного входного примера выведите формулу для $\nabla_{W^{(l)}} ce$ — градиента оценочной функции по весам $W^{(l)}$, используя $\delta^{(l)}$. Сначала выведите формулу для одной компоненты, затем для всего вектора.

9. По аналогии с предыдущим пунктом, для батча примеров выведите формулу для $DW[l] = \nabla_{W^{(l)}} CE$ через $DZ[l] = \nabla_{Z^{(l)}} CE$. Сначала выведите формулу для одной компоненты, затем для всей матрицы в векторизованном виде. Выпишите размеры всех матриц.

10. Выпишите все формулы обратного прохода для батча в векторизованном виде. Для этого необходимо выразить $DW[l] = \nabla_{W^{(l)}} CE$ через матрицы $X, Y, \hat{Y}, A[l], Z[l]$, с помощью выведенных в предыдущих пунктах рекуррентных соотношений и вычисляя $DZ[l] = \nabla_{Z^{(l)}} CE$ в качестве промежуточных значений. Чтобы ускорить обратный проход, старайтесь переиспользовать матрицы выходов, активаций и предактиваций, вычисленные на прямом проходе. Рядом с каждой формулой выпишите размеры всех матриц. Чтобы упростить процесс реализации, рекомендуем выводить размерности всех вычисляемых матриц и сравнивать с выписанными.

Примечание. Для удобства векторизации рекомендуется поддерживать число строк (axis=0) матриц Z, A, DZ равным размеру батча. Если вы испытываете сложности с выполнением этого задания, финальные формулы, которые необходимо вывести, можно посмотреть в слайдах лекций. Попробуйте сначала вывести формулы для отдельных элементов матриц DZ, DW, а затем уже векторизовать их вычисление. Внимание! Для учета смещения (bias) формулы в презентации требуется модифицировать (добавить операцию, обратную конкатенации 1 на прямом проходе)!

Практическая часть

Реализуйте нейронную сеть с L слоями (L-1 скрытым слоем) и обучить ее на датасете FILIMDB. Ответы на поставленные вопросы сдайте в составе письменной части.

A Загрузите и распакуйте матрицы эмбеддингов **GloVe** отсюда. Для отладки и ответа на вопросы используйте эмбеддинги размерности 50, для финального обучения выберите оптимальные эмбединги (50, 100, 300).
Вопрос A.1: Покажите, что косинус угла между векторами совпадает с их скалярным произведением, если вектора предварительно нормировать (поделить на евклидову норму). Выведите формулу, выражающую евклидово расстояние через косинус угла между

векторами для нормированных векторов.
Вопрос A.2: Выберите 10 слов, начинающихся с первых двух букв вашей фамилии в латинской транскрипции. Для выбранных вами слов найдите 15 ближайших слов. Сравните результаты при использовании в качестве меры близости скалярного произведения и косинуса угла между векторами – для этого разместите их в соседних столбцах таблицы. В каждой ячейке таблицы приведите исходное слово, ближайшие слова, отсортированные по убыванию меры близости, и значения меры близости.
Вопрос A.3: Найдите 50 пар максимально близких друг к другу слов:

$$\arg \max_{w_i, w_j : i<j} sim(w_i, w_j) \tag{1}$$

Сравните результаты при использовании в качестве меры близости скалярного произведения и косинуса угла между векторами. Приведите сами пары, отсортированные по убыванию меры близости, и значения меры близости. *Примечание.* Самая эффективная реализация сводиться к умножению матрицы эмбедингов на саму себя транспонированную EE^T (получаем матрицу близости каждого слова к каждому) и выбору top k элементов из этой матрицы. Однако при этом получится матрица, которая не влезет в память. Простое решение — посчитать top k ближайших соседей для каждого слова по очереди: $E[i]E^T$, затем слить результаты и выбрать из них top k ближайших пар. Для более эффективного решения можно разбить словарь на батчи по 100-1000 слов. Изучите функции np.ravel, np.partition и подумайте, как их зайдествовать для решения задачи.
Вопрос A.4: Визуализируйте эмбеддинги всех слов, начинающихся с первых двух и со вторых двух букв вашей фамилии в латинской транскрипции (например, Иванов выбирает из словаря все слова, начинающиеся с iv и с va). Для визуализации воспользуйтесь методами снижения размерности и постройте диаграмму рассеивания (scatter plot). Для каждого эмбеддинга на графике добавьте подпись соответствующего слова. Для снижения размерности можно воспользоваться методом анализа главных компонент (Principal Component Analysis, PCA) из sklearn (см. пример). Выполняются ли свойства дистрибутивной семантики? Какие слова группируются в кластеры? *Примечание.* Для визуализации можно воспользоваться библиотекой matplotlib (см. совсем краткое и чуть более подробное руководства по ней). Рисовать графики можно прямо в Jupyter Notebook, если правильно его настроить (см. пример).

B Напишите функцию предобработки и токенизации отзывов. Обратите внимание, что в следующем пункте, мы будем искать для каждого токена его GloVe эмбеддинг. Соответственно, предобработка текста должна выдавать токены в том виде, в котором они хранятся в GloVe словаре. Посмотрите, что именно нужно сделать с текстом (нужен ли перевод всех букв в нижний регистр, есть ли там знаки препинания, цифры?) и реализуйте соответствующую предобработку и токенизацию.

C Напишите функцию векторизации отзыва. Для этого для каждого токена из отзыва вам нужно найти его GloVe эмбеддинг, и в качестве векторного представления отзыва выдавать среднее арифмитическое от GloVe эмбеддингов всех токенов из отзыва.

D Преобразуйте обучающую и тестовую выборки в матрицы размера $N * d$, где N – количество отзывов в выборке, а d – выбранный вами размер эмбеддингов **GloVe** (50, 100 или 300). Каждая строка представляет собой вектор, полученный при векторизации отзыва из выборки с помощью функции векторизации из предыдущего пункта. **Вопрос D:** Для скольких токенов и какого количество уникальных слов из обучающей и тестовой выборки вы не нашли GloVe эмбеддинги? Приведите 20 примеров таких слов.

E Опираясь на формулы из теоретической части задания, реализуйте прямой проход для нейронной сети с L-скрытыми слоями. Размер скрытых слоев вы можете выбрать произвольно. Рекомендуемое стартовое количество скрытых слоев: 2, стартовое значение размера скрытых слоев: 200,100. Для реализации прямого прохода, вам потребуется имплементировать следующие функции:

$relu(z)$, $tanh(z)$, $sigmoid(z)$ - функции активации;

$init\_params(layer\_sizes, activation)$ принимает список размеров слоев сети и строку с именем функции активации (sigmoid/relu/tanh/linear); возвращает начальные значения весов в виде словаря с ключами - именами весов и значениями - соответствующими матрицами (используйте Xavier initialization / He initialization в зависимости от функции активации, подробнее тут)

$fully\_connected(a\_prev, W, activation)$ принимает выход предыдущего слоя, веса текущего слоя, строку с именем функции активации (sigmoid/relu/tanh/linear); возвращает выход текущего слоя и кэш промежуточных значений, которые потребуются на обратном проходе;

$ffnn(X, params, activation)$ принимает батч примеров, веса сети и строку с именем функции активации; возвращает $Z^{(L)}$ - предактивации последнего слоя и список кэшей, полученных из функции $fully\_connected$;

$softmax\_crossentropy(ZL, Y)$ принимает предактивации последнего слоя и матрицу one-hot векторов классов, возвращает значение оценочной функции и кэш. Если ZL содержит большие положительные компоненты, после их экспоненцирования может возникнуть переполнение. Чтобы этого избежать, вычтите из каждой строки максимальное значение в этой строке, воспользовавшись свойством softmax, доказанным в теоретической части.

Вопрос E.1: Сразу после случайной инициализации чему равно матожидание $\hat{y}(x)$? Чему в среднем равно матожидание оценочной функции без регуляризации? Вычислите значение оценочной функции без регуляризации на обучающей выборке, чему оно равно?

Вопрос E.2: Постройте гистограммы, показывающие, как распределены компоненты входных векторов в зависимости от слоя нейросети, а также графики изменения среднего значения и дисперсии в зависимости от слоя. Для построения распределений воспользуйтесь функцией hist (см. пример), а для графиков зависимости среднего и дисперсии от номера слоя функцией – plot из библиотеки matplotlib (см. пример). Как меняются распределения с увеличением номера слоя? Попробуйте уменьшить и увеличить начальные случайные веса в 100 раз и постройте такие же графики. Как они изменились? Какой эффект оказывает на обучение сети инициализация весов слишком маленькими или слишком большими значениями и почему?

F Опираясь на формулы из теоретической части задания, реализуйте обратный проход для нейронной сети с L-скрытыми слоями. Для реализации обратного прохода, вам потребуется имплементировать следующие функции:

$fully\_connected\_backward(dA, cache, activation)$ принимает градиент оценочной функции по выходам текущего слоя, кэш и строку с именем функции активации (sigmoid/relu/tanh/linear); возвращает градиент по выходам предыдущего слоя и градиент по матрице весов текущего слоя;

$ffnn\_backward(dZL, caches, activation)$ принимает градиент по предактивациям последнего слоя, список кэшей и строку с именем функции активации; возвращает словарь с ключами - именами параметров сети и значениями - градиентами по этим параметрам;

$softmax\_crossentropy\_backward(cache)$ принимает на вход кэш и возвращает градиент по предактивациям в последнем слое.

G Корректная реализация функции обратного прохода может быть довольно сложной задачей, в которой очень легко допустить ошибку. Для проверки вашей реализации мы **настоятельно** рекомендуем вам провести **gradient checking**. Подробнее про gradient checking вы можете прочитать здесь. **Вопрос G:** Проведите gradient checking на каждом слое нейронной сети. С какой точностью сходятся градиенты, посчитанные численно, с градиентами, полученными из backpropagation модуля?

H Реализуйте функции обучения сети:

$sgd\_step(params, grads, learning\_rate)$ - делает шаг градиентного спуска, обновляя веса сети из params в направлении, противоположном градиенту по ним из grads;

$train\_ffnn(Xtrain, Ytrain, Xdev, Ydev, layer\_sizes, learning\_rate, num\_epochs, batch\_size)$ обучает сеть с указанными гиперпараметрами, возвращает выученные параметры сети,

а также списки значений оценочной функции и точности на обучающей и валидационной выборках в конце каждой эпохи.

I Для проверки работоспособности и отладки написанной сети можно:

1) создать сеть без скрытых слоев и убедиться, что полученная линейная модель дает результаты, сопоставимые с логистической регрессией из домашнего знадания 2 при использовании BOW-векторов;

2) добиться того, что сеть переобучится на небольшом батче.

J Установите learning rate равным 1e−2, коэффициент L_2 регуляризации α равным 1e−5. Обучите нейронную сеть на матричном представлении обучающей выборки и разметьте с их помощью тестовую выборку. ? **Вопрос J.1:** Нарисуйте два графика, показывающих изменение оценочной функции и точности классификатора в процессе обучения (графики обучения), на каждом графике нарисуйте кривые для train (метрики на каждом батче) и dev (метрики на полном dev в начале каждой эпохи). Через сколько эпох обучение сходится (оценочная функция перестает меняться)? **Вопрос J.2:** Какой точности классификатора вам удалось достичь на обучающей и тестовой выборке? Имеет ли место переобучение классификатора или недообучение? Что нужно сделать с α, чтобы улучшить результат?

K Аналогично заданию с логистической регрессией проведите оптимизацию гиперпараметров learning rate и коэффициента L_2 регуляризации α на валидационной выборке. **Вопрос K:** Чему равно оптимальное значение α? Какое потребовалось число эпох и learning rate для обучения до сходимости?

L Используя найденное в предыдущем пункте оптимальное значение гиперпараметров α и learning rate, обучите классификатор на всей обучающей выборке. **Вопрос L.1:** Оцените чему равна точность классификатора на обучающей и тестовой выборке? **Вопрос L.2:** Сколько времени занимает обучение классификатора и предсказание результатов для тестовой выборке?

Исследовательская часть

1. Попробуйте улучшить точность классификации с помощью нейронной сети, поэкспериментировав с ее архитектурой (количество и размер слоев). Нарисуйте график зависимости точности от 1) размера слоя, 2) количества слоев. Какую оптимальную архитектуру вам удалось подобрать?

2. Здесь вы можете ознакомится со списком доступных эмбеддингов. Сравните точность, которую вам удается достичь с помощью **Word2vec** эмбеддингов, с точностью которой вы уже достигли на **GloVe** эмбеддингах.

3. Попробуйте при обучении сети считать градиенты и обновлять эмбеддинги слов. Это может помочь выучить эмбеддинги, которые лучше описывают релевантные задаче признаки слов (тональность).

4. Как показывает статья "Deep Unordered Composition Rivals Syntactic Methods for Text Classification", можно добиться дополнительного прироста точности, используя подход Word Dropout. Идея состоит в том, чтобы избежать переобучения с помощью дополнительной регуляризации: в процессе обучения нейронной сети мы для каждого слова будем подбрасывать монетку и с некоторой вероятностью p выкидывать его из текста. В итоге, мы получим обучающую выборку гораздо большего размера, в которой присутствует множество версий текстов с разными отброшенными словами. Такая аугументация текста приводит к увеличению точности и большей устойчивости обученной нейронной сети.

5. Помимо классического градиентного спуска, существуют различные его вариации, такие как Adam, Adagrad, RMSProp, Momentum которые активно используются в реальном

мире. Суть этих подходов состоит в том, что для каждого следующего шага градиентного спуска помимо непосредственно градиентов, учитывается момент (вторая производная). Подробный обзор методов здесь. Попробуйте реализовать Momentum или Adagrad и использовать его для обучения нейронной сети из практической части.

6. Вместо усреднения эмбеддингов слов для построения эмбеддинга предложения можно использовать их взвешенное среднее. В качестве весов можно взять значения весов наивного байеса, tf-idf или реализовать стратегию из "A Simple but Tough-to-Beat Baseline for Sentence Embeddings". Попробуйте реализовать один или несколько из предложенных подходов.