Машинное обучение (Machine Learning)

Сеть Хопфилда, ограниченная машина Больцмана, рекуррентная нейронная сеть

Уткин Л.В.

Санкт-Петербургский политехнический университет Петра Великого

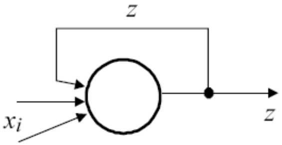
Содержание

Презентация является компиляцией и заимствованием материалов из замечательных презентаций и материалов по машинному обучению:

Eric Jang, Ferenc Huszar, Camron Godbout, Christopher Olah, Oleksandr Sosnovshchenko, Denny Britz

Сеть Хопфилда

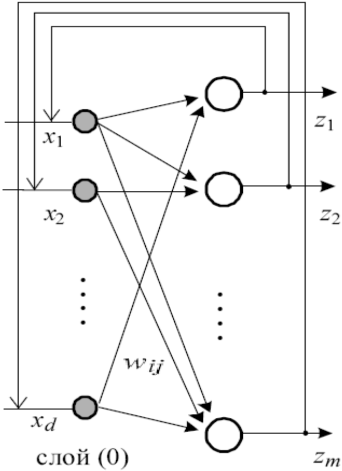
Нейрон с обратной связью



1. На вход нейрона подаются входные значения x_j и вычисляется выход нейрона z.
2. Затем z подается на вход нейрона наряду с прочими значениями и вычисляется новое выходное значение z.
3. Этот процесс повторяется до тех пор, пока выходное значение нейрона будет мало изменяться от итерации к итерации.

Устойчивая сеть Хопфилда

В 1982 г. Хопфилд предложил устойчивую рекуррентную биполярную $\{-1, 1\}$ сеть



Устойчивая сеть Хопфилда

- Только один слой настраиваемых весов w_{ij}
- Все нейроны единственного слоя возвращают свои выходы на свой вход и входы всех остальных нейронов сети посредством распределителей (не нейронов) слоя (0)
- Каждый нейрон реализует следующие шаги:
    1. вычисляет взвешенную сумму своих входов:

$$a_j = \sum_{i \neq j}^{M} (w_{ji} z_i) + x_j$$

    2. к сумме применяется нелинейная пороговая функция

$$z_j = g(a) = \begin{cases} 1, & a_j > T_j, \\ -1, & a_j < T_j, \\ \text{не меняется}, & a_j = T_j. \end{cases}$$

Устойчивость сети Хопфилда

Сеть гарантированно устойчива при выполнении условий:

1. матрица весов W симметрична $w_{ij} = w_{ji}$;
2. имеет нули на главной диагонали $w_{ii} = 0$(нет обратных связей)

- Рекуррентная сеть - динамическая система, имеющая энергетическое состояние
- Энергия E- мера близости к стабильному состоянию (функция Ляпунова):

$$E = -\frac{1}{2}\sum_i \sum_j w_{ji} z_j z_i - \sum_j x_j z_j + \sum_j T_j z_j$$

- T_j - порог нейрона j
- Любое изменение состояний сети уменьшает энергию системы, и сеть Хопфида является устойчивой

Ассоциативная память

Человеческая память является ассоциативной: мозг воспринимает какую-то информацию (например имя человека) и в ответ возвращает целую гамму воспоминаний (внешность, место, эмоции и т.п.), то есть, задавая некоторую часть информации, получаем всю остальную.

Сети Хопфилда могут формировать упрощенную модель ассоциативной памяти

Ассоциативная память - алгоритм

1. Все запоминаемые образы x_j, $j = 1, ..., M$, кодируются биполярными векторами длины N (нейронов).
2. Веса сети Хопфилда настраиваются:

$$w_{ij} = \sum_{d=1}^{M} x_d^{(i)} x_d^{(j)}, \quad x_d = (x_d^{(1)}, ..., x_d^{(N)})$$

3. Восстановление ассоциаций: Входам придают значение образа, возможно частично искаженного, и сеть колеблется до своего устойчивого состояния и стабилизируется в одном из запомненных состояний. Значения выходов - восстановленная ассоциация.

Ассоциативная память

Ограниченная машина Больцмана

Машина Больцмана

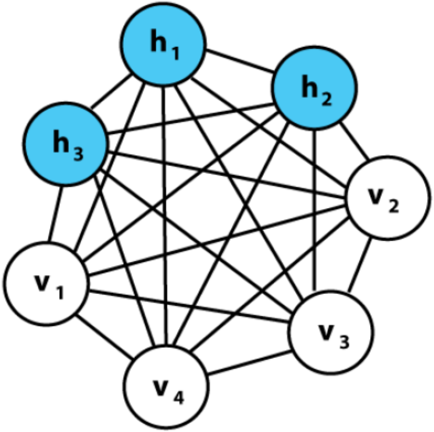
Нейронная сеть является стохастической, если ее веса (связи между нейронами) принимают случайные значения, либо функция активации нейрона является случайной. В последнем случае нейронную сеть называют также ***машиной Больцмана (Restricted Boltzmann Machine - RBM)***.

RBM *- полносвязанный неориентированный граф, где нейроны поделены на две группы, описывающие* ***обозреваемые*** *и* ***скрытые*** *состояния*

Машина Больцмана

- Это двухслойная сеть с порождающими возможностями
- У них есть способность узнать распределение вероятностей по его набору входных данных
- Могут использоваться для уменьшения размерности, классификации, регрессии, совместной фильтрации

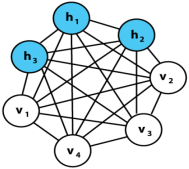
Машина Больцмана



Машина Больцмана

- Машины Больцмана являются стохастическими порождающими моделями глубокого обучения только с двумя типами узлов - hidden и visible узлы.
- Нет выходных узлов! Это может показаться странным, но это то, что дает сетям стохастическую особенность.
- У них нет типичного выхода типа 1 или 0, через который шаблоны изучаются и оптимизируются с использованием Stochastic Gradient Descent.
- Они учат модели без этой способности, и это делает их такими особенными!

Машина Больцмана



- Все узлы связаны со всеми остальными узлами независимо от того, являются ли они входными или скрытыми узлами.
- Это позволяет им обмениваться информацией между собой и самостоятельно генерировать последующие данные.
- Измеряем только то, что находится на видимых узлах, а не то, что на скрытых узлах.

Энергия совместной конфигурации сети

- Связи между нейронами: $w_{ij} = w_{ji}$; $w_{ii} = 0, \forall i$.
- Энергия:

$$E(v,h) = -\frac{1}{2}\sum_i \sum_j w_{ij} v_i h_j - \sum_j a_j v_j - \sum_j b_j h_j$$

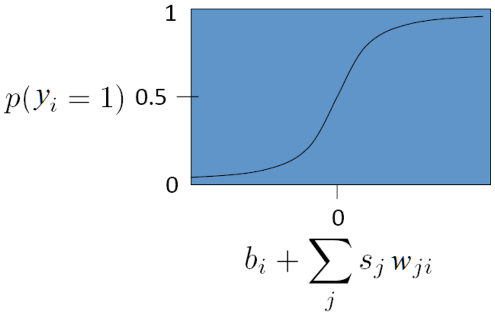
v_i, h_i - состояния видимого и скрытого нейрона
a_i, b_i - смещения видимых и скрытых нейронов

$$-\frac{\partial E(v,h)}{\partial w_{ij}} = v_i h_j$$

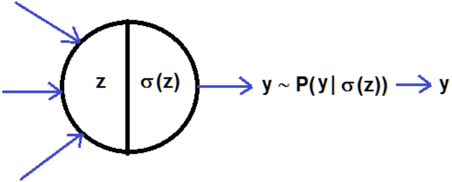
Стохастический двоичный нейрон

На выходе: 0 или 1, но эти значения не определяются однозначно взвешенной суммой входов, а зависят от нее стохастически; вероятность появления 1 на выходе нейрона

$$P(y_i = 1) = \frac{1}{1 + \exp(-b_i - \sum_j s_j w_{ji})}$$



Стохастический нейрон



$$P(y_i = 1) = \frac{1}{1 + \exp(-\sum_j s_j w_{ji}/T)} = \frac{1}{1 + \exp(-\Delta E_i/T)}$$

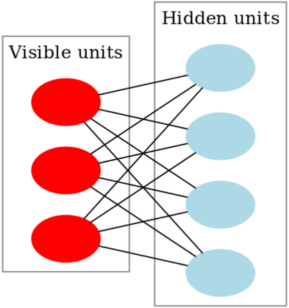
$$\Delta E_i = E(y_i = 0) - E(y_i = 1)$$

T - аналог температуры, используемый для управления степенью неопределенности

Ограниченная машина Больцмана

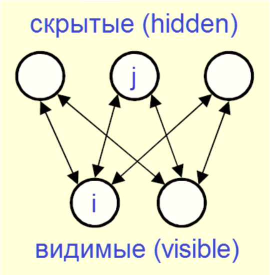
- Особый класс машин Больцмана
- Ограничены с точки зрения связей между видимыми и скрытыми нейронами, что облегчает их реализацию по сравнению с машинами Больцмана
- Это двухслойная нейронная сеть (один слой - видимый слой, а другой - скрытый слой), и эти два слоя связаны полностью двудольным графом
- Каждый узел в видимом слое связан с каждым узлом в скрытом слое, но никакие два узла в одной группе не связаны друг с другом

Ограниченная машина Больцмана



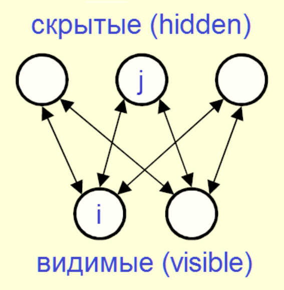
Ограниченная машина Больцмана

- Ограничим связи, чтобы сделать **обучение проще**
- Если убрать связи внутри группы, чтобы получился двудольный граф, получим структуру модели RBM
- При данном состоянии нейронов одной группы, состояния нейронов другой группы будут независимы друг от друга

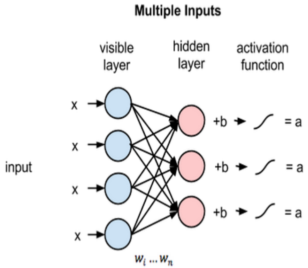


Плюсы структуры

- Только один слой скрытых нейронов
- Можно быстро поучить несмещенную выборку из апостериорного распределения на скрытых нейронах ("причинах"), когда есть вектор данных



Обучение (1)



- Это первый шаг в обучении RBM с несколькими входами
- Входные данные умножаются на веса и затем добавляются к смещению
- Затем результат передается через сигмоид, и вывод определяет, активируется ли скрытое состояние или нет
- Весами будет матрица (число входных узлов X число скрытых узлов)

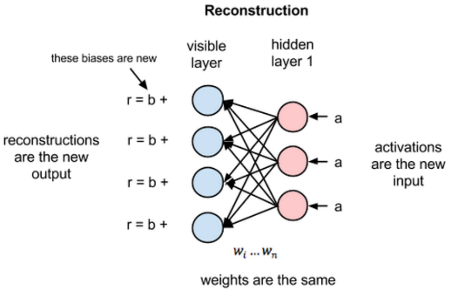
Обучение (2)

- Уравнение на этом этапе

$$\mathbf{h}^{(1)} = \sigma\left(\mathbf{v}^{(0)} W + \mathbf{a}\right),$$

где $\mathbf{h}^{(1)}$ и $\mathbf{v}^{(0)}$ - векторы (матрицы столбцов) для скрытого и видимого слоев с верхним индексом в качестве итерации ($\mathbf{v}^{(0)}$ - вход сети)

Обучение (3)



- Это обратная фаза или реконструкция
- Похожа на первый проход, но в противоположном направлении

Обучение (4)

- Уравнение: $\mathbf{v}^{(1)} = \sigma\left(\mathbf{h}^{(1)}W + \mathbf{a}\right)$, $\mathbf{v}^{(1)}$ вектор смещения видимого слоя
- Разница $\mathbf{v}^{(0)} - \mathbf{v}^{(1)}$ - ошибка реконструкции, которую нужно уменьшить на последующих этапах обучения
- Т.о., веса корректируются на каждой итерации, чтобы минимизировать эту ошибку
- Это и есть обучение.

А теперь вероятности

- В прямом проходе вычисляется вероятность выхода $\mathbf{h}^{(1)}$, учитывая вход $\mathbf{v}^{(0)}$ и вес W: $p\left(\mathbf{h}^{(1)}|\mathbf{v}^{(0)}; W\right)$
- В обратном проходе при реконструкции входа вычисляется вероятность выхода $\mathbf{v}^{(1)}$, учитывая вход $\mathbf{h}^{(1)}$ и вес W: $p\left(\mathbf{v}^{(1)}|\mathbf{h}^{(1)}; W\right)$
- Веса, используемые как в прямом, так и в обратном проходе, одинаковы
- Вместе эти две условные вероятности приводят к совместному распределению входных данных и активаций: $p(\mathbf{v}, \mathbf{h})$

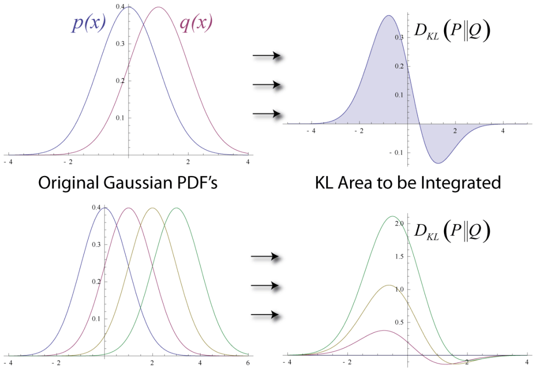
Порождающее обучение

- Реконструкция отличается от регрессии или классификации тем, что оценивает распределение вероятностей исходного ввода вместо того, чтобы связывать непрерывное / дискретное значение с примером ввода
- Это означает, что делается попытка угадать несколько значений одновременно
- Это - **порождающее обучение**

Снова вероятности

- Как алгоритм уменьшает потери или уменьшает ошибку на каждом шаге?
- Пусть есть два нормальных распределения, одно из входных данных $p(x)$ и одно из восстановленного входного приближения $q(x)$.
- Разница между распределениями и есть ошибка в графическом смысле и надо минимизировать ее, то есть максимально приблизить графики.
- Дивергенция Кульбака-Лейблера (KL-дивергенция)
- Алгоритм оптимизации RBM пытается минимизировать KL, изменяя веса так, чтобы реконструкция очень напоминала входные данные.

KL-дивергенция



Веса -> Энергия -> Вероятности

- Каждая возможная совместная конфигурация скрытых и видимых нейронов имеет “энергию” Хопфилда (определяется весами и смещениями).
- Энергия совместной конфигурации скрытых и видимых нейронов определяет вероятность того, что сеть выберет эту конфигурацию.
- Управляя энергиями совместных конфигураций, можно управлять вероятностями, которые модель назначает видимым нейронам (это дает очень простой и эффективный алгоритм обучения).

Вероятностное описание RBM

RBM вычисляет совместную вероятность пар v и h:

$$p(v, h) = \frac{1}{Z} e^{-E(v,h)}$$

Z - параметр нормализации, при наличии n_1 образцов v, и n_2 - образцов h:

$$Z = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} e^{-E(v^{(i)},h^{(j)})}$$

Полная вероятность $p(v)$ конфигурации видимых нейронов v (функция активации состояний видимого слоя) - сумма по всем h:

$$p(v) = \sum_{j=1}^{n_2} p(v, h^{(j)}) = \frac{1}{Z} \sum_{j=1}^{n_2} e^{-E(v,h^{(j)})}$$

Вероятностное описание RBM

$$p(v, h) = \frac{1}{Z} e^{-E(v,h)}$$

Интерпретация $p(v, h)$: инициализировав видимый слой v вектором из обучающего множества, и, вычислив значения скрытого слоя h на основе текущего видимого, возможно посчитать вероятность $p(v, h)$ текущего состояния системы.

Вероятностное описание RBM

Функция активации состояний скрытого слоя: вероятность того, что образ v, поданный на вход сети, содержит признак k:

$$h_k \sim P(h_k = 1|v) = \frac{e^{-E_1}}{e^{-E_1} + e^{-E_0}} = \frac{1}{1 + e^{-b_k - \sum v_i w_{ik}}}$$
$$= \sigma\left(-b_k - \sum v_i w_{ik}\right).$$

Так как при данном v все h_k не зависят друг от друга, то вероятность текущего состояния

$$P(h|v) = \prod_{k=1}^{n_2} P(h_k = 1|v)$$

Вероятностное описание RBM

Функция активации состояний видимого слоя: вероятность того, что образ h, поданный на вход сети, содержит признак j:

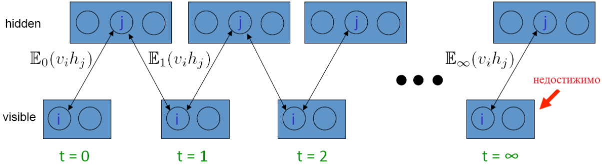
$$v_j \sim P(v_j = 1|h) = \frac{1}{1 + e^{-b_j - \sum h_i w_{ij}}}$$
$$= \sigma\left(-b_j - \sum v_i w_{ij}\right).$$

Вероятность текущего состояния

$$P(v|j) = \prod_{j=1}^{n_2} P(v_j = 1|v)$$

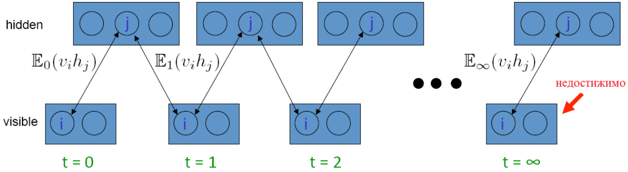
Обучение RBM

- В процессе обучения вычисляются вероятности $P(h_k = 1|v)$ для вектора состояний $h^{(k)}$ на основе текущего значения вектора состояний $v = v^{(k)}$.
- После данного этапа вычисляются вероятности $P(v_j = 1|h)$ для вектора состояний $v^{(k+1)}$ из полученных ранее значений $h^{(k)}$, до тех пор, пока нейронная сеть не “восстановит” вектор $v^{(0)}$.
- “Восстановление” - получение вектора $v^{(n)}$, максимально точно описывающего изначально поданный на вход вектор $v^{(0)}$.



Обучение RBM

- Полная вероятность конечного вектора $v^{(n)}$ равна $p(v)$ (определена ранее).
- Цель обучения - сделать так, чтобы восстановленный вектор был наиболее близок к оригиналу, т.е. максимизировать $p(v)$
- Метод - частные производные вероятности по параметрам w_{ij}, a_j, b_j



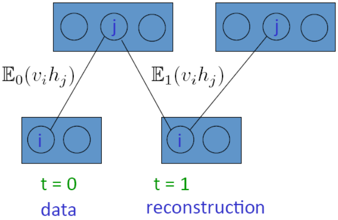
$$\frac{\partial \log p(v)}{\partial w_{ij}} = \mathbb{E}_0(v_i h_j) - \mathbb{E}_\infty(v_i h_j)$$

Интересный факт

Все, что один вес должен знать о других весах и данных для того, чтобы максимизировать $p(v)$ содержится в разности двух корреляций

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \mathbb{E}_0(v_i h_j) - \mathbb{E}_\infty(v_i h_j)$$

Быстрое обучение



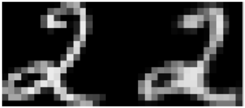
- Начинаем с обучающего вектора на видимых нейронах
- Модифицируем все скрытые нейроны
- Модифицируем все видимые нейроны, чтобы получить реконструкцию
- Модифицируем все скрытые нейроны снова

$$\Delta w_{ij} = \epsilon \left[\mathbb{E}_0(v_i h_j) - \mathbb{E}_1(v_i h_j)\right]$$

Пример обучения

Данные

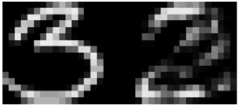
Реконструкция из активированных двоичных признаков



Новые тестовые рисунки из класса, на котором модель уже обучилась

Данные

Реконструкция из активированных двоичных признаков



Рисунки из незнакомого класса (сеть пытается увидеть каждый рисунок как 2)

Общая информация о рекуррентных нейронных сетях

История и особенности

- RNN были созданы в 1980-е, но только сейчас стали популярны благодаря созданию мощных графических процессоров и развитию NN.
- Используются при работе с последовательной информацией — в основном с текстами и аудио/видео-сигналами.
- Традиционные нейронные сети не имеют памяти и не совсем ясно, как знания о предыдущих событиях могут помочь классифицировать последующие события.

Особенности функционирования

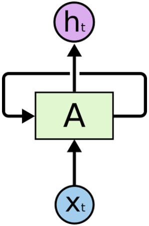
- В RNN каждый нейрон взаимодействует сам с собой, т.е. использует свою внутреннюю память, чтобы сохранять информацию о предыдущем входе.
- Благодаря этому, фразы “I had washed my house” и “I had my house washed” могут различаться.На вход RNN как правило передается сигнал, являющийся некоторой последовательностью.
- Каждый элемент такой последовательности поочередно передается одним и тем же нейронам, которые свое же предсказание возвращают себе вместе со следующим ее элементом, до тех пор пока последовательность не закончится.

Параметры и данные

- x_t - вход на шаге t, например, x_1 может быть вектор, соответствующий второму слову в предложении.
- A_t - скрытое состояние в момент t. Это “память” сети. A_t вычисляется на основе предыдущего скрытого состояния: $A_t = f(UA_t + WA_{t-1})$, $A_{-1} = 0$.
- h_t - выход на шаге t, например, если хотим прогнозировать следующее слово в предложении, h_t - вектор вероятностей, определенный на множестве слов словаря. $h_t = \mathrm{soft}\max(VA_t)$

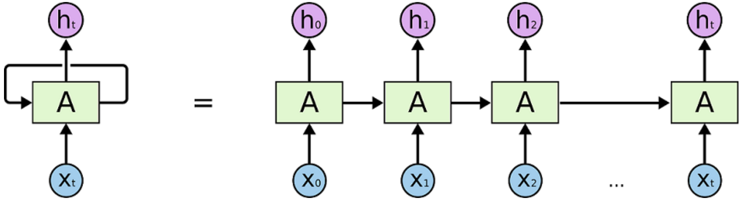
Представление элемента

- Элементы рекурентной сети изображают как обычные нейроны с дополнительной циклической стрелкой, которая демонстрирует то, что кроме входного сигнала x_t и выхода h_t нейрон использует также свое дополнительное скрытое состояние A.



Развернутое представление элемента

- Если “развернуть” такое изображение, получится цепочка одинаковых нейронов, каждый из которых получает на вход свой элемент последовательности, выдает предсказание и передает его дальше по цепочке как своего рода ячейку памяти.
- Это абстракция, поскольку это один и тот же нейрон, который отрабатывает несколько раз подряд.

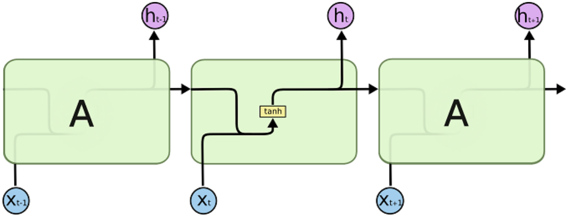


Простейшая последовательность данных

- Пусть нейронная сеть получает на вход последовательность данных, например, текст пословно или слово побуквенно.
- Каждый следующий элемент этой последовательности поступает на нейрон в новый условный момент времени.
- К этому моменту в нейроне уже есть накопленный с начала поступления информации опыт.
- В фразе «в ясном небе светит солнце» в качестве x_0 выступит вектор, характеризующий предлог "в", в качестве x_1 - слово "небе" и так далее. В итоге в качестве h_t должен быть вектор, близкий к слову "солнце".

Внутренняя структура нейрона

- Основное отличие разных типов рекурентных нейронов друг от друга кроется в том, как обрабатывается ячейка памяти внутри них.
- Традиционный подход подразумевает сложение двух векторов (сигнала и памяти) с последующим вычислением активации от суммы, например, гиперболическим тангенсом.
- Получается обычная сетка с одним скрытым слоем.



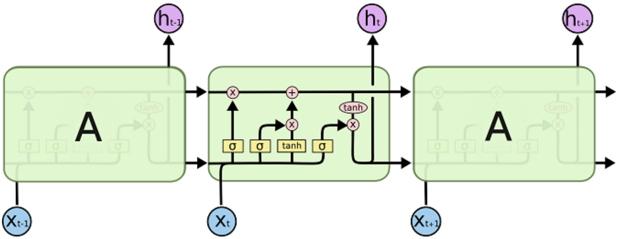
Проблема организации

- Память, реализованная подобным образом, получается весьма короткой. Поскольку каждый раз информация в памяти смешивается с информацией в новом сигнале, спустя 5-7 итераций информация уже полностью перезаписывается.
- Если обрабатываемый текст длинный, то закономерности в его начале уже не будут вносить какой либо вклад в решения сети ближе к концу текста.
- Это проблема исчезающего градиента.

Сети долго-краткосрочной памяти

LSTM-RNN

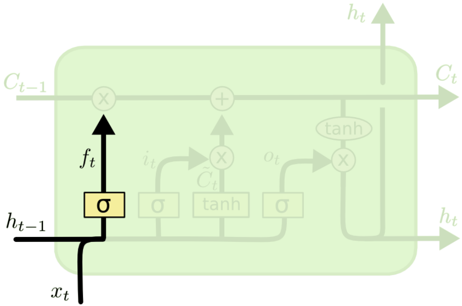
Long Short-Term Memory Recurent Neural Network (LSTM-RNN): добавлены дополнительные внутренние преобразования.



Первый слой

- На первом шаге LSTM решает, какую информацию нужно отбросить (забыть) из ячейки.
- Решение принимается сигмоидом, называемым "вентиль забывания".
- Он "смотрит" на h_{t-1} и x_t, и выдает число между 0 и 1 для каждого числа в ячейке C_{t-1}.
- 1 - "полностью оставить это", 0 - "полностью отбросить"

Первый слой

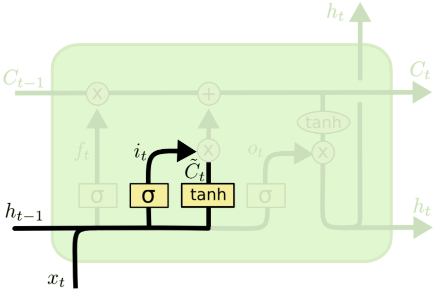


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Второй слой

- Второй слой вычисляет, насколько ему интересна новая информация, чтобы запоминать ее.
- Он имеет две части:
  - первая - сигмоидальный слой, называемый "входной вентиль", решает какие значения будут модифицированы.
  - вторая - слой tanh создает вектор значений нового кандидата $\tilde{C}_t$, который мог быть добавлен к ячейке.
- На следующем шаге эти две части комбинируются для модификации состояния.

Второй слой

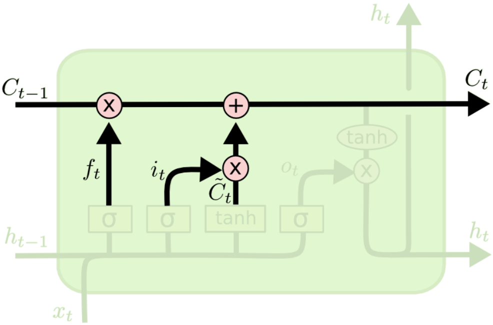


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Модификация состояния

Новое состояние памяти C_t - линейная комбинация памяти C_{t-1} и наблюдения $\tilde{C}_t$ с только вычисленными весами для каждой из компонент:

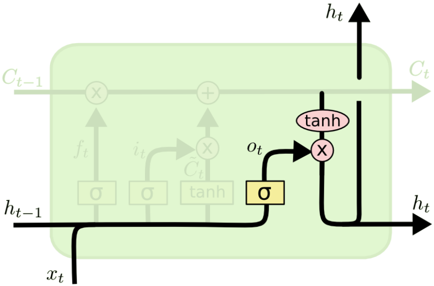


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Финальный шаг - вычислить output

- Так как часть входного сигнала уже в памяти, не нужно считать активацию по всему сигналу.
- Сначала сигнал проходит через сигмоиду, которая решает, какая его часть важна для дальнейших решений.
- Затем гиперболический тангенс “размазывает” вектор памяти на отрезок от -1 до 1.
- В завершение, эти два вектора перемножаются.

Вычисление output



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Полученные h_t и C_t передаются далее по цепочке.

Варианты LSTM-RNN

- Различные варианты реализации LSTM-RNN сети можно найти здесь:
  - Greff K., Srivastava R.K., Koutnik J., Steunebrink B.R., Schmidhuber J. LSTM: A Search Space Odyssey // arXiv:1503.04069v1, Mar 2015.
  - Jozefowicz R., Zaremba W., Sutskever I. An Empirical Exploration of Recurrent Network Architectures // Proc. of the 32-nd Int. Conf. on Machine Learning, France, 2015
  - http://alexsosn.github.io/ml/2015/11/16/LSTM.html
  - http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Обучение LSTM-RNN

- Обратное распространение по времени (backpropagation through time)
- Функция потерь - кросс-энтропия:

$$E(o, \widehat{o}) = \sum_t E_t(o_t, \widehat{o}_t) = -\sum_t o_t \log \widehat{o}_t$$

- o_t - сорректное слово на шаге t, $\widehat{o}_t$ - предсказанное.
- Обычно фраза рассматривается как один обучающий пример. Поэтому общая ошибка равна сумме ошибок на каждом шаге времени (слове).
- Используется стохастический градиентный спуск.

Где целесообразно применять?

- Аннотация картинок
- Создание музыки
- Классификация протеинов
- Генерация человеческого почерка
- и во многих других задачах...

Ресурсы

Описания (почти одинаковые):

- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://medium.com/@camrongodbout/recurrent-neural-networks-for-beginners-7aca4e933b82#.564cf0419
- https://habrahabr.ru/company/dca/blog/274027/
- http://www.kdnuggets.com/2015/06/rnn-tutorial-sequence-learning-recurrent-neural-networks.html

Программное обеспечение:
RNN в R: package '**rnn**'

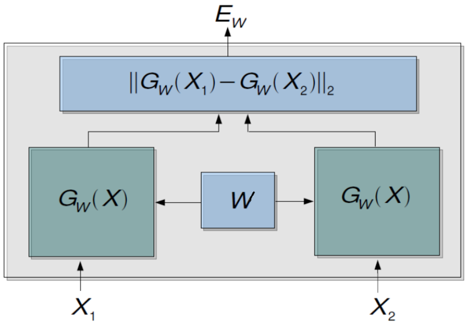
Сиамские нейронные сети

Элементы сиамских сетей

- X_1 и X_2 - пара изображений
- $Y = 0$, если X_1 и X_2 - один объект, $Y = 1$, если X_1 и X_2 - различны
- Построить нейронную сеть с минимальным числом параметров, определяющую для пар объектов, одинаковы ли она или нет

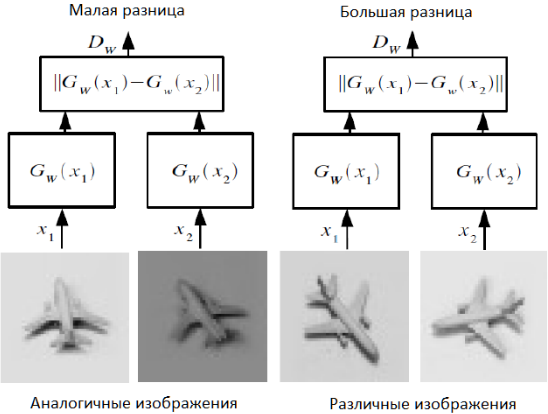
Архитектура сиамских сетей

Y. LeCun. Learning Hierarchies of Invariant Features



- W - общий вектор параметров,
- $G_W(X_1)$, $G_W(X_2)$ - точки в прост-ве меньшей размерности
- E_W - функция совместимости между X_1 и X_2 (“энергия”)

Еще пример сиамских сетей



Y. LeCun. Learning Hierarchies of Invariant Features

Функция потерь

- Функция потерь зависит от входных данных и параметров косвенно через энергию:

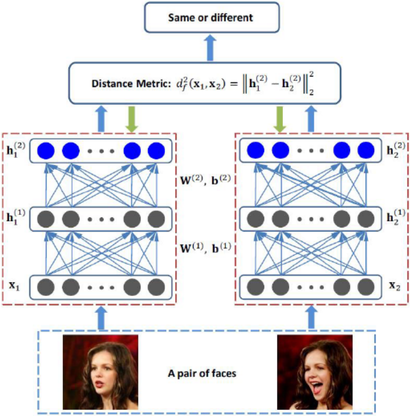
$$\mathcal{L}(W) = \sum_{i=1}^{N} L\left(W, (Y, X_1, X_2)_i\right)$$

$$L\left(W, Y, X_1, X_2\right) = (1-Y) L_G\left(E_W(X_1, X_2)\right) + Y L_I\left(E_W(X_1, X_2)\right)$$

$$E_W = \|G_W(X_1) - G_W(X_2)\|$$

- L_G - функция потерь для совпадающих пар $Y = 0$
- L_I - функция потерь лоя несовпадающих пар $Y = 1$

Применение к распознаванию лиц



JunlinHu, etc. Discriminative Deep Metric Learning for Face Verification in the Wild, CVPR 2014

Вопросы

?