

Избранные направления в нейронных сетях

Часть 1. Как обучать нейронные сети

Topic 1. Of Theory to Practice (6-38)

1.1 Революция

we need to go deeper

1994: нейронные сети - это хороший способ сделать практический шаг

Хикон, середина 2000х: преобразование Тесла венчурного бизнеса хорошо и без него, этикета и традиций

Второй разброс: большие физики. Множество (2006 - именем GPU)
модель MNIST → 1,2 Тб ImageNet

1.2 Искусственный интеллект и машинное обучение

→ Amazon Mechanical Turk - 6 часть шахматного автомата "Deep Blue",
разработанного Кеннеди I

Начало ИИ связано с Яковиным: М.М., архитектура, прибл. 20, 20
помогает себе разобраться. Компьютер не имеет и.
Чтобы быть умным нужно быть за кого-то сеть воровать

1956 - Берклиский семинар

Автоматический логический вывод (automated theorem proving)
, эпоха искусственного интеллекта, перспектива не имеет

1970-е - knowledge-based systems

к началу 1980-х - back propagation

Вторая волна увлечения, одна из самых обещающих ??

Было известно, что strong AI может появиться при новых алгоритмах
TODO: вспоминать источники

1.3. Классы ошибках машинного обучения

- Обучение без учителя
 - Обучение нейросети (определить распределение)
 - Снижение размытости (так, что можно \neq бесс. исключения)
 - Класификатор
- Обучение с частичной привлечением учителя
(использовано, если правило ясно, а неравнозначно)
- Обучение с учителем
 - Класификатор
 - Регрессия
 - Обучение разрывами
- Обучение с подкреплением (напр. при решении игр A/B) ??

1.4. Особенности человеческого мозга

Мозг работает параллельно в зависимости от задач взаимодействия нет синхронизации (когда мы, например, зеваем дважды в синхронии)

Распознавание лиц делает ученые из <10% людей, некоторые

т.е. мозг очень параллелен и при этом добавлено массы - GPU, где CPU

есть много "горизонтальных" слоев, слоев между нейросетевыми единицами etc. Мозг не линейный

Нейронные сети не пытаются имитировать человеческий мозг!
Они спроектированы для быстрого обработки для решения оптимизационных задач

Параллельность мозга - есть специализированное участки, но нейроны не всегда параллельны (различные роли!), обрабатывают разные нейроны и синапсы

Есть ли "единий алгоритм" обучения человеческого мозга? ??

1.5. Презентация вопросами: что мы не можем дать роботу?

"Может ли нейронная сеть наука?" *Митч Калк
Процессор MOS 6502 3510 транзисторов, 1.5 ГГц за секунду вычисляет
много задач, которые невозможно
сделать человеком, кроме небольших, общих изображений

1.6. База и модели современных ИИ

Многие хороших примеров уже требуют человека (распознавание лиц, игр). В чем отличие от концепции Тьюринга, например?
Во-первых человека можно было бы обучить: числа, мн-ва, новому
и т.д. Второе: части:

1. Историческая физика: крайне приближенные фиг. модель мира, способная к очень малым обобщениям и переходу на новые виды выхода
2. Историческая психология. Например, изучение Марии Черепановой даёт понять, что так и неиз

Во-вторых, люди хороши в переносе обучения (Transfer Learning)

Могут повторять абстракции из очень небольшого числа примеров
типа: яблоко через one-shot learning

В-третьих, причинность (causality) – способность воспользоваться, используя
"принцип" происходящего. Например, научившись различать
яблоки, хорошо учим чайки, и улучшаем эти навыки

Лекция 2. Предварительные сведения | 38-93
или курс машинного обучения

2.1. Частота

В моделях неубедительной неопределенности, важно её учитывать, а
именно, нужно учить ее определять

- дискретное с. фн., как или ск. число исходов, p в сумме 1
- напр. с. фн., p -это распределение $F(a) = p(x < a)$, плотность p -фн.: $p(x) = \frac{dF}{dx}$, $\int p(x) dx = 1$
- совместная вероятн. двух событий: $p(x,y)$. независимые величины, если $p(x,y) = p(x) \cdot p(y)$
- условная вероятн. $p(x|y) = P(x,y)/p(y)$. условная вероятн.: $p(x,y|z) = p(x|z) \cdot p(y|z)$

Th Bayes (Байес). По оп. вер. вер-сю $p(x,y) = p(x|y) \cdot p(y) = p(y|x) p(x)$

$$\Rightarrow p(y|x) = \frac{p(x|y) p(y)}{p(x)} = \frac{p(x|y) p(y)}{\sum_{y' \in Y} p(x|y') p(y')} - \text{Ф-ла или Th Bayes}$$

Нужно ли непрерывное априорное представление о мире ($p(y)$)

на основе частной информации от наблюдения ($p(x|y)$)

и уже бывшая получаса состоящие наших представлений ($p(y|x)$)

Прямая задача: из модели оценить вер-сю происшествие в реальности

Обратная задача: по исходном наблюдении оценить вер-сю модели

(по новейшим статистическим методам построить вероятностную модель)

Еще вер-сю можно & как степень уверенности (когда речь о со-дополнительных неизв.)

Th Bayes (в терминологии ML)

$$p(\theta|D) = \frac{p(\theta) \cdot p(D|\theta)}{p(D)} = \frac{p(\theta) \cdot p(D|\theta)}{\sum_{\theta} p(D|\theta) p(\theta) d\theta}, \text{ где}$$

• $p(\theta)$ - априорная вер-сю (prior probability)

• $p(D|\theta)$ - правдоподобие (likelihood)

• $p(\theta|D)$ - апостериорная вер-сю (posterior probability)

• $p(D) = \int p(D|\theta) p(\theta) d\theta$ - вероятность данных (evidence)

Чтобы подобрать θ

→ в макс. статистике ищут инф. max likelihood: $\theta_M = \arg \max_{\theta} p(D|\theta)$

→ в байесовском подходе и обр. ML ищут max апостериорного инф.

$$\theta_{MAP} = \arg \max_{\theta} p(\theta|D) = \arg \max_{\theta} p(D|\theta) p(\theta)$$

Можно подобрать апостериорный (и угодный для вычислений) инф.,
оценить likelihood и получить posterior

Чтобы предсказать, нам нужно предсказуемое распределение
 $p(y|D)$ или $p(y|D, x)$, где y -слег. пример в данных или ответ на вопрос x

$$p(y|D) = \int_{\theta} p(y|\theta) p(\theta|D) d\theta \underset{\substack{\text{апостериорное} \\ \text{распределение}}}{\approx} \int_{\theta} p(y|\theta) p(\theta) p(D|\theta) d\theta$$

В классе МЛ оптимизируют по параметрам θ_{MAP} или θ_{ML}

$$\theta_{MAP} = \arg\max_{\theta} p(\theta|D) = \arg\max_{\theta} p(D|\theta) p(\theta) = \left[\begin{array}{l} \text{точек в данных} \\ \text{нормализовано} \\ \text{независимо} \end{array} \right]$$

$$= \arg\max_{\theta} p(\theta) \prod_{d \in D} p(d|\theta) = \left[\begin{array}{l} \text{переходы} \\ \text{нормировано} \end{array} \right] = \arg\max_{\theta} \log p(\theta) + \sum_{d \in D} \log p(d|\theta)$$

В первом оптимизируют $\log p(D|\theta)$ - лог. правдоподобие
и $\log p(\theta)$ - регуляризатор

2.2. Градиентные методы и регуляризация

На слова про град. сходимость: $\nabla_\theta E = \left(\frac{\partial E}{\partial \theta_1}, \dots, \frac{\partial E}{\partial \theta_n} \right)^T$

$$\theta_t = \theta_{t-1} + u_t, \text{ где } u_t = -\gamma \nabla_\theta E |_{\theta=t-1}$$

Минимизация критерия: find posterior $(p(\theta) p(D|\theta))$ обычно
используют $RSS(\theta) = \sum_{i=1}^k (y_i - x_i^\top \theta)^2$ - это есть аналитическое решение оптим.

Чему RSS ? Это max likelihood в предположении о нормальном шуме

Регуляризация: L_2 вспоминает об рисках о с.норм. распределении для θ

2.3. Рассстояние Кульбака-Лейблера и перекрестная энтропия

Ассиметричный критерий \Rightarrow град. быст. не помогает

Рассстояние Кульбака-Лейблера (относительная энтропия) - мера нехожности
двух вероятностных распределений $P \ll Q$. Определяется кол-вом информации,
которое требуется приближения P с помощью Q

$$\text{напр.: } KL(P||Q) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)}$$

$$\text{напр.: } KL(P||Q) = \int_{\mathbb{R}^d} p(x) \log \frac{p(x)}{q(x)} dx$$

Перекрестная энтропия. $H(p, q) = E_p[-\log q] = -\sum_y p(y) \log q(y)$

П.к. будим для bin. кн. истинное расп. $p(y=1) = y, p(y=0) = 1-y$, то

$$\text{это минимальный критерий } L(\theta) = H(p_{\text{истин.}}, q(\theta)) = -\frac{1}{N} \sum_{i=1}^N y_i \log \hat{y}_i(\theta) + (1-y_i) \log (1-\hat{y}_i(\theta))$$

Н.З.: задать бессмыслицно!

Логистическая регрессия (одинаково на последние и первые слова)

$$\text{тут будет } \ln \frac{y}{1-y} \text{ или разные аналоги. } -\sum_{n=1}^N [t_n \ln y_n + (1-t_n) \ln(1-y_n)]$$

не счастье разобралась тут №3: карточки с определением? Э. ПО

2.4. Градиентный спуск: основы

Скоро научимся брать производную от целевой функции по весам.

Но она же векторная \rightarrow мы хотим, чтобы максимизировать, более того, у нас есть масса ограничений, что и масса — обратите внимание! Эвристические методы решают!

Математическое описание: если с производной не очень, нужно минимизировать

1. приближенное вычисление производной
2. производная в форме, приближающей исходную
3. производная в градиентном \rightarrow дополнительную форму ошибки

25. Градиентный спуск и градиент на нем

Представляем стоимость функции как композицию более простых элементарных из которых функций, где хотим минимизировать ее саму и ее производную по другому аргументу

Хотим минимизировать стоимость функции ошибки (loss)

- $f \circ g$ — скомбинированные функции $\Rightarrow (f \circ g)'(x) = (f(g(x)))' = f'(g(x)) \cdot g'(x)$
- f — векторная, g — скаляр. $\nabla_x f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \Rightarrow \nabla_x(f \circ g) = \begin{pmatrix} \frac{\partial f \circ g}{\partial x_1} \\ \vdots \\ \frac{\partial f \circ g}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x_n} \end{pmatrix} = \frac{\partial f}{\partial g} \nabla_g g$
- обе векторные, $\Rightarrow \nabla_x f = \frac{\partial f}{\partial g}, \nabla_x g, \dots, \frac{\partial f}{\partial g_k}, \nabla_x g_k = \sum_{i=1}^k \frac{\partial f}{\partial g_i} \nabla_i g_i$:
то же, что иначе: $\nabla_x f = \nabla_g g \nabla_g f$, где $\nabla_g g = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \dots & \frac{\partial g_k}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial x_n} & \dots & \frac{\partial g_k}{\partial x_n} \end{pmatrix}$ — матрица $k \times k$ единиц

• мы можем комбинировать можно применять
6 разных направлений:
• от нейронов к строкам, получая чистые выходы. А выходы из
нейронов и это же переменной = выходу • от строк к нейронам, получая
чистые строки из всем тренирующимися

• в M2 лучше 2-й вариант — это обычный loss, который есть на базовом
• Это и есть back propagation, т.е. в противоположном направлении

Глава 3. Перцептрон или эмбрион будущего компьютера

* boom-and-boost cycles

21:40 -

3.1 Когда появляют искусственные нейронные сети

- Идея о сумматоре + сравнении с порогом — никакого обсуждения, просто модель
- Идея о вспышках и исчезновении связей → обсуждение по ходу (всегда)
- Идея о нейронах (после изучения биологических нейронов)

3.2. Как работает перцептрон

здесь приблизительно

* перцептрон Розенблата — линейная модель классификации (дискретной)

У входа $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, на выходе $y(x) \in \{-1, 1\}$

Нужен веса $w_0, w_1, \dots, w_d \in \mathbb{R}$, чтобы отрез $\text{sign}(w_0 + w_1 x_1 + \dots + w_d x_d)$ делит пространство на $y(x)$

задаваем $x_0 = 1$, то есть можно просто склонять $w^T x$

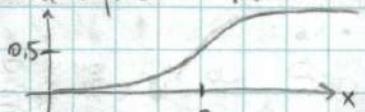
Выводим формулу ошибки: хотим минимизировать без разрывов

Фундаментальный критерий перцептрона: $E_p(w) = -\sum_{x \in M} y(x)(w^T x)$, где M — все ошибки

Правило обучения Розенблата: если на входах x_1, x_2, \dots , и если ошибки, обозначим $w^{(t+1)} = w^{(t)} - \eta \nabla_w E_p(w) = w^{(t)} + \eta z_n x_n$

Когда ввести критерий активации, иначе нейронную сеть не построить. Классический вариант — логистическийsigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Действительно непрерывно.
 $\lim_{x \rightarrow 0} \sigma(x) = 0$, $\lim_{x \rightarrow \infty} \sigma(x) = 1$

?! Как следствие, менять форму ошибки, теперь будет перекрестная энтропия

$$E(w) = -\frac{1}{N} \sum_{i=1}^N (y_i \log \sigma(w^T x_i) + (1-y_i) \log (1-\sigma(w^T x_i)))$$

Будем, теперь моделировать их в сети и обучать SGD!
 организуем нейроны в слоях во имя бесконечности

Задача И.И.: начали разработание в машинном переводе,
 кроме и что перцептрон избыточно скользко-то грустных
 выражений, что оттолкнуло от этой темы исследователей

3.3 Современные нейронные синапсы активации

Название	Формула $f(x)$	Формула $f'(x)$	Комментарии
Логистич. сигмоид σ	$\frac{1}{1+e^{-x}}$	$f(x)(1-f(x))$	(0, 1), более гладкая, касающаяся линейной тангенс \tanh
Гипербол. (q, Xebuage)	$\begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	$1 - f^2(x)$	$(-1, 1)$ тоже более гладкая, $\sigma'(0) = 1/4$, расчет и подобие тангенса: $\tanh'(0) = 1$
ReLU rectified linear units	$\begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	$\begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	0-само воспроизводит точка перехода норм на конечном уровне обнуляет производную
Sigmoid	$\log(1+e^x)$	$\frac{1}{1+e^{-x}}$	<ul style="list-style-type: none"> один экспоненциальный фактор \rightarrow один фактор стек. подчиняется (из (1)) - бесполезно приводит к узкокодному нейрону (2) исключение, сумма ∞ для дополнительно, так и σ
LeLU (логарифмическая)	$\begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$	$\begin{cases} a, & x < 0 \\ 1, & x \geq 0 \end{cases}$	<ul style="list-style-type: none"> a - положительное > 0 (без а) 2 вида производных при $x < 0$ и $x \geq 0$
ELU (exponential)	$\begin{cases} ale^{-1}, & x < 0 \\ x, & x \geq 0 \end{cases}$	$\begin{cases} f(x)+a, & x < 0 \\ 1, & x \geq 0 \end{cases}$	<ul style="list-style-type: none"> стремительно близко к нулю стремительно близко к единице постоянность

Несложные виды производных приходят в игру - σ и ReLU

- (1) Учёв: склоняю ко сумме - сложениея ступенек, ктн разница между $\sigma(5)$ и $\sigma(10)$. Чтн если построить $f(x) = \sigma(x+0.5) + \sigma(x-0.5) + \sigma(x-1.5) + \dots$? Получим что-то похожее на $\log(1+e^x)$, чтн похоже на ReLU
- (2) В може склоняю разрешимость бкн. нейронов $(1-4\%)$
 $\sigma \Rightarrow 50\%$. ReLU с правильной регуляризацией - как \log

3.4. Как же обусловить исходящие нейроны

Не через grad. фунции: - где тута упаковки никак не передать
 - броско было член - или. 0/1
 - нет градиент. связей

Обусловие по Хэдзу: веса \uparrow там, ктн сюди исполнуются

\rightarrow Сети Хонфинга: "заполняют" несколько активаций, они становятся лок. мин., есть из начального состояния \uparrow к ним

?! Регуляризующий активаций по нейронной сети - склоняю
 Вроде такой способ обусловления работает, но медленнее, чем grad. фунции
 Всё, забыли про пропорц., & нейронные сети как аддитивные

3.5 Чудесные сети: в чём прелесть и в чём опасность?

Что же такое, но получше большое развитие Таблица 6 2005-2006

А.И. Иванченко, метод группового учёта органическ. соединений, кото
 рый я и сам, исп. вонгаго. проф. как входы сплош.

Зачем вообще чудесные н.с.? Они более эффективно предсказывают,
 даже если число нейронов остаётся постоянным.
 Свой нейронов с ReLU-активацией фактически "ворачивает" пр-во
 "одномерного некоторое это части между собой": и поэтому различаются
 поверхности, построенные в этом "свернутом" пространстве, потому
 "разворачиваются" в гораздо более сложные конструкции в пр-ве
 содержимого входных сигналов

Чему же тогда они подались так хорошо?

→ проблема удаляющихся градиентов (vanishing gradients) - последние шли обучаться быстро и хорошо, но после него проходящие через них градиенты обнуляются

→ проблема взрывающихся градиентов (exploding gradients) в кэшур. слоях

В середине 2000-х придумали предобучать сети без учителя, чтобы они знали что-то о нр-ве до оптимизации и не складывались быстрее в пок. мин. Но оказалось, что это не абсолютное

Появились различные инструменты регуляризации и оптимизации (просто градиент, деснект, нормализация мин-бакс, ини-циз, различные сгл...). Плюс позже стало сильно получше

3.6. Пример распознавания рукописных цифр на TensorFlow

“MNIST – фотография машинного обучения” – Примеры Хилтон

Класс 4 быстрее, чище, сильнее
или DB евклидов, дельных и гранитных

4.1 Регуляризация в нейронных сетях

Буквальное ограничение на размер весов

→ L₂ регуляризатор: $\lambda \sum w^2$ Ridge

→ L₁ регуляризатор: $\lambda \sum |w|$ Lasso

В keras есть kernel_regularizer (матрица весов сети), bias_4.. и activity_4.

Train-test split, метод early stopping, борюлько, αL_2

Деснект – статистика-Модифицированное. Это в некотором смысле р, что это включает в себя сеть работает как обычно, но боятся майонеза. $p = 0.5$ подходит. Почему работает? Вероятно, как аналог среднего моделей в один архитектуре. Деснект позволяет нейронам работать независимо, наклоняясь картине. Тогда в меру все тоже работает способом

4.2 Как минимизировать веса

CSG

- Чем дальше отходит нейрон от себя тем лучше. Автоматизировано
- ? • В древних революционных прошлых алгоритм contrastive divergence, это можно исправить настолько, что значение как правило. Но это последовательно глубокое минимум бывает рано
- То же, но с автоматизированием на steepest - stacked (минимизацией) autoencoders

Принципы предобучки:

1. последовательно, от нижних слоев к верхним. Решает проблему vanishing gradients
 2. без учёта, т.е. параметра не изменяется
 3. в результате получается модель, которая соотносится с реальностью
- Но самое деле сейчас мало используется

Инициализация весов (для симм. функций активации)

- прямой ход \times дисперсия значений активации в слое нейронной сети
[Есть симметричный ход, потому что $y = w^T x + b$, и для несимм. функции активации]
получаем, что дисперсия выходов \approx дисперсия входов
Если инициализированном распределением, дисперсия в k слое делится на 3, что приводит к ухудшению инициализации]
 - обратное распределение – аналогичная ситуация
 - итог: для беспрепятственного распространения значений активации и градиента по сети дисперсия в обоих случаях ≈ 1
 - Это невозможно сделать однозначенно для неравнозначных размеров слоев, поэтому берется определенное распределение
- TODO эксперименты на keras \Rightarrow и правда стабильно лучше случайное инициализации!

Задача про симметрию (у которой нет слоев)

Он не очень хорошо для глубоких сетей, т.к. близко находимся в локальных минимумах только начинаем учиться, и выход последнего слоя не имеет связи с выходом сети, сеть может находиться в месте const – средние значения выходов. Для этого нужно инициализировать так, $\sigma(-\infty) \rightarrow 0$ и $\sigma'(-\infty) = 0$, застрянем в этом положении.

Также это может происходить, но это сложнее, особенно с хорошей инициализацией.

Инициализация χ_e (для несимм. функций активации)

В keras: he_uniform и he_normal

Виды:

- для симм. функций активации – иниц. каскад (tanh в основном)

• для несимм. (tanh) – иниц. χ_e

? норм/нормализован? – видимо, как параметр при подборе

4.3 Нормализация по мини-батчам

Нормализовать мини-батчей

1. усреднение градиента по батчу аппроксимирует градиент по X , но в большинстве сильно лучше
2. одновременная обработка начали примеров - GRU

Проблема внутренних сдвигов переменных: если между батчами сильно меняется распределение входов нейронов, то это уже вогнутое веса будут давать худшее. К тому же для симметрии это надо не делать.

Одни из основных методов решения - какая-либо нормализация. Входных данных или входов слоя (с помощью специального отдельного слоя)

Этот слой ходит по всему нейрону X для E и Var , но приходится ограничить батчом. Но нужно параметризовать зан. степеньми свободы γ_k и β_k , чтобы можно было, если нужно, настраивать тонк. функцию

? (при $\gamma_k = \sqrt{\text{Var}[x_k]}$ и $\beta_k = E[x_k]$)

$$y_k = \gamma_k x_k + \beta_k = \gamma_k \frac{x_k - E[x_k]}{\sqrt{\text{Var}[x_k]}} + \beta_k$$

Целевой слой нормализации по мини-батчу $B = \{x_1, \dots, x_m\}$

1. Вычисление базовые статистики: $\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$; $\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$

2. Нормализует входы: $\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
3. Вычисляет результат: $y_i = \gamma \hat{x}_i + \beta$

Несколько советов: делать ли нормализацию после мин. части (до функции активации) или уже после всего слоя

Чтобы эксперимент на MNIST, слои 784, 100, 100, 10 $\xrightarrow{\text{тут нормализация}}$ помогает

Плюс нейроп. сейчас у параметров β и γ добавляется разница в времени по времени по переменным, в том числе довольно большим

Еще небольшой вопрос: есть ограничение размера батча, чтобы были адекватные статистики

! Следующий шага огромна, а у меня Рутин и команда работают

? Как они выбирали подбатчики к шагам?

4.4. Метод моментов: Ильюшин, Несколов и Гессе

Вспомним про параметр град. спуска γ . Естественно его можно динамизировать.

Часто используется линейное затухание: $\gamma = \gamma_0 \left(1 - \frac{t}{T}\right)$ или эксп.: $\gamma = \gamma_0 e^{-\frac{t}{T}}$
Но параметров стало больше, инос. поверхность лосса не учитывается

Аддитивные методы град. спуска учитывают происходящее с функцией

Метод импульсов: чистое движение в одном направлении, уменьшает колебание (пример с узким образом). Эмулируем инерцию точки
 $u_t = \gamma u_{t-1} + \gamma \nabla \phi E(u)$ $\gamma = 1 - \eta$ $\eta < 1$ - доля инерции в рез.

Метод Несколова: смотрим на поправку вперед, чтобы не споткнуться о камень

$$u_t = \gamma u_{t-1} + \gamma \nabla \phi E(u - u_{t-1})$$

Метод Ильюшина: используется не только ' γ ', но и ' α ' (= метод бывшего положения)

т.е. приближаем нашу функцию не линей, а параболой.

$[H(E(u))]$ - гессиан или матрица Гессе - матрица производных 2 порядка

$$u = -[H(E(u))]^{-1} \nabla E(u) \quad \text{Это быстрее и скорость настраивается сама}$$

но на практике бывает что H генерируется

В рез-те всё равно приходится как-то уменьшать γ по ходу дела

- поправка уменьшается: $\gamma \approx 1$ раз в N эпох
- уменьшить, когда прекращает уменьшаться ошибки на $validat$
- trade-off между вычислительным бюджетом оптимума и скоростью обучения

4.5 Аддитивные варианты градиентного спуска

до сих пор учитывали только текущий градиент и γ , но не историю обновления для конкретного параметра, а мб он уже близок к оптимальному

Аддитивные методы сами это учитывают, хороши на рулев., дают и более стабильную

? Аддукад: шаг изменения для меньше у тех параметров, которые в большей степени варьируются в данных и наоборот

используется G_t - диагональная матрица из сумм квадратов градиентов соответствующего параметра за предыдущие шаги: $G_{t,ii} = G_{t-1,ii} + g_{t,i}^2$

Элементы матрицы и есть скорость обновления отдельных параметров.

Главный минус - скорость пон. 1. порядка слишком медленно

Adadelta - небольшое модификации

1. сумму квадратов не копи, а считаю по окну либо с экспоненциальными весами
2. использующие матрицы лессе, вместо рутинности (матрица)

RMSprop - образ-блужданий из курса Хитона (без модификации), немного другой

Adam - тоже модификация Adadelta, изменяющие веса ср. и грб. градиентов:

$$m_t = \beta_1 m + (1 - \beta_1) g_t \quad v_t = \beta_2 m + (1 - \beta_2) g_t^2 \quad \hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

т.е. Adam Optimizer. Но градиент сходится сильно быстрее, хотя и с большой дисперсией

Практические советы

- скорее всего Adam будет хорошо, но если нет, пробовать другие
- изменение скорости по расписанию добавлять распределение
- важно следить за ошибкой на validation и бояться ее увеличиваться
- работать на train, validate (параметр, остановка) и test (finalное качество)

Глава 5. Сверточные нейронные сети и архитектуры, или Не верь глазам своим

5.1 Зрительная кора головного мозга

Сверточные сети (Convolutional NN, CNN) - перенаправляет атри и те же части сети для работы с различными маленькими, локальными участками изображений

Части зрительной коры:

- V1 - локальное признаки небольшого участка
- V2 - проектирует лок. признаки, чтобы обобщить и добавить бинокулярность
- V3 - цвет, геометрия, первые результаты сегментации и группировки
- V4 - геом. рисунок и отражение простых объектов. результаты внимания
- V5 - движение: направление и скорость объектов из V4
- V6 - обобщение движущихся картинок, учет движений человека
- V7 (?) - распознавание сложных объектов, в т.ч. человеческих лиц

Разделение, наконец на четкую сеть нейронной сети. Наибольшее есть связи непропущены между V2 и V5, между промежуточными. Возможно, внимание из V4 возвращается к V1 и V2. Это в МЛ пока мало реализовано (будет в 8.1)

Интересно, что кейропт в ИИ расположено в соответствии с "карточкой".

Критик 1. локальная структура отражена хорошо
 2. недавно - мало изменений
 3. за 2% новых данных в центре отбирают 50% кейроптов
 4. есть исключения, наклонение на преобразование к поларным координатам

На что влиуются кейропты в ИИ кроме расположения?

- ориентацию \rightarrow границы изображений (edge detection)
- пространственная частота - как часто меняется освещенность?
- направление в пространстве и времени (через анимацию)
- различие между каналами (хотя большинство обрабатывается только один)
- цвет по одному из трех направлений: красный-зеленый, синий-желтый, ч-б

See also: фильтр Габора про ориентацию,

SIFT (scale-invariant feature transform) - как активируются (ключевые точки)

Критик в кое & в зоне тоже еще несколько слоев!

5.2 Свертки и сверточные сети ?! Ради этого?!

Линейное преобразование - non-linear ($Wx + b$)

Но иногда есть информация о внутренней структуре данных, напр., каналы изображения

Целк: получаем вход маленькими окнами (зб 5x5 пикселей) и в & будем применять линейную сеть. Их выход можно трактовать как картинку etc

В ИИ пикселе входного изображения есть текстур с каналами (если речь о входных данных) либо карта признаков (если уже обработали сверточки). Карта может меняться

Для свертки - линейное преобразование особого вида

Если X -карта признаков с шагом e , то результат двумерной свертки

с ядром размера $2d+1$ и матрицей весов $W \in M(2d+1, 2d+1)$

$$X = \begin{pmatrix} 0 & 1 & 2 & 1 & 0 \\ 4 & 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 & 1 \\ 1 & 2 & 3 & 1 & 0 \\ 0 & 4 & 3 & 2 & 0 \end{pmatrix} \xrightarrow{\text{шаг свертки } e \text{ на уровне } d} \begin{pmatrix} 2 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \xrightarrow{\text{шаг весов } W} \begin{pmatrix} 2 & 0 & 1 \\ 1 & 2 & 3 \\ 0 & 4 & 3 \end{pmatrix} \xrightarrow{\text{шаг весов } W} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} \xrightarrow{\text{шаг весов } W} \begin{pmatrix} 9 & 5 & 4 \\ 8 & 8 & 10 \\ 8 & 15 & 12 \end{pmatrix} = X * W$$

Свойства свёртки:

- сохраняет структуру входа
- разрешена с операцией, т.к. А неизменяется от небольшого числа единиц предшествующего слоя (зв. в полносвязной - от всех)
- свёртка многократно переносится и не теряет веса

CSG

Ночти всегда после свёртки искажается линейность:

$$z_{ij}^c = h(y_{ij}^c), \text{ часто ReLU, особенно в очень глубоких сетях}$$

Переход от полносвязного слоя к свёрточному:

1. добавляет предположение о локальности признаков, сразу с явной параметрической входом
2. для локальной ячейки сети дает радиообразные входы окрестности, т.к. из одной исходной картинки получаем только изображение
3. проблема: не видно связей между узлами пикселами (\rightarrow грубые!)

за это вина

картинка \leftarrow каналы

После свёртки и линейности обычно идет субдискретизация (aka pooling, subsampling, подвыборка): обобщает информацию о наиважнейших признаках, "забывает" о точном расположении

(это стандартный свёрточный слой):

1. свёртка (линейное отображение), т.к. линейная сеть лок. признаки
2. локальная линейность
3. субдискретизация

Элементы свёртки на фильтрах. Обычно 3×3 , но и 1×1 для свёрток по каналам (зв. кадрам бифер)

<тут пример для MNIST>

Early stopping по качеству на validate

NB: почему мы пишем свёртки хорошими фильтрами?

5.4 Современные сверточные архитектуры

2014 VGG (Visual Geometry Group) — одна из самых популярных сверточных архитектур
основное изобретение — фильтры 3×3 с шагом 1
вместо использовавшихся ранее 7×7 с шагом 2 и 11×11 с шагом 4

- Аргументы:
1. Результирующее поле трех слоев 3×3 равно 7×7 , но есть у них общих 27 против 49 в слое 7×7 \rightarrow можно сжимать!
 2. Большие нейроны \Rightarrow большие "разрешающая способность"
(аналогично, если смотреть в фильтрах 1×1)

Особенности:
(изображение сначала)

- но 2-3 сверточные слои получают уже не столько фильтров, сколько уже сжатые
- чем дальше тем больше выходных feature maps
- в конце все сжимаются в вектор и \rightarrow полносвязное

Inception (by Google)

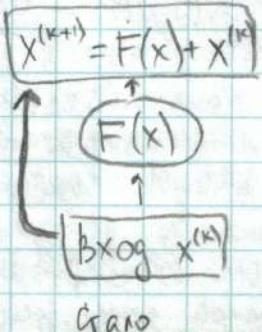
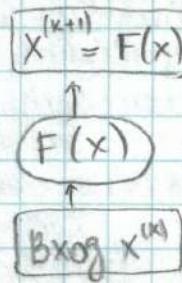
- Стартовые блоки: модули, комбинирующие сверточки 1×1 , 3×3 , 5×5 + max pooling
- Блок — объединение 4 маленьких сетей, их выходы объединяются
- Вспомогательные: 1×1 используются для нейронов, связанных для размежевания
- Число для обучения 22 параметризованых ячеек (проблема затухания градиентов):
в среднем сети включают нейроны из промежуточных признаков, чтобы выходы не могли влиять на loss (-0.3). Скорость не улучшается, но регуляризация делает лучше

Следующая версия: все "большие" сверточки $\rightarrow 3 \times 3$ и $1 \times n$

1. $n \times n$ заменяются на последовательные $1 \times n$ и $n \times 1$ (также \sim сингуларному разложению матрицы)
2. Все $5 \times 5 \rightarrow 2$ послед. 3×3

В результате удаётся обучать глубокие сети без затухания градиентов
Только вот проблема — маленькие сети лучше по качеству, ч
это не переобучение, т.к. не train тоже. Сложнее обучать глубокие

2015 Глубокое остаточное обучение (Deep residual learning) of Microsoft
Resnet. Идея: есть "остаточная сеть" напрямую от входа слоя к выходу



Понимается, блок должен аппроксимировать исходную $H(x)$, а остаток $H(x) - x$

Проблема: из-за отсутствия баланса, нет проблем с затуханием градиентов, но лучше без него

Можно добавлять слои, но лучше без них

Сейчас в большинстве задач используются глубокие нейронные сети ResNet, особенно с картинками

Раньше скорость (зБ на мобильном) можно было MobileNets и SqueezeNet,

2015 Работы изменили всё: искусственные сети (highway net.)

$$y^{(k)} = C(x^{(k)})x^{(k)} + T(x^{(k)})F(x^{(k)}), \text{ обычно } C=1-T$$

C - линия переноса (саччу gate), T - преобразование (transform)

Это конструкции тоже быстро отдаются, но не такие, как глубокие сети

5.5. Автомоделирование

Решают задачи извлечения признаков, обучение без учителя

Хотим обучить функцию $f(x, w) \approx x$ где w - параметры

Суть задачи - какие ограничения наложить на внутренние слои

Простой вариант: сделать внутренний слой залито линиями по размерности, с амплитудой R^d не получится это, но наше мн-во же-то неподдаётся от многообразию линий по размерности

Классические решения затемнение размерности

- PCA (principal component analysis), SVD (singular value decomposition)
 - ICA ("independent -ness")
- ко вторичные сети шире и淺нее

Оказывается, что при затемнении размере скрытых слоев (overcomplete autoencoders) легче перебрать лучше, чем при затемнении (undercomplete a.e.)

Часто контролять не встает - неизвестность, регул-ческ, понял.

Изумительный способ добавить с переобучением - шумоподавляющий автомоделировщик (denoising) $f(\tilde{x}, w) = x$, где \tilde{x} - зашумленный x

- Бесплатно \Rightarrow обучаемый блокорук дополнительная регуляризация как шуметь:
 - $+ N(0, \sigma^2)$, σ^2 маленькая
 - зашумить часть входов (для картинок)

Размеренные ак. - регуляризует зома активированных нейронов
заряжая расстоянием до истинной зомы: $KL(P, \hat{P}) = P \log \frac{P}{\hat{P}}$

также разрешим:

- идея / idea / concept / principle / беса можно удалить \rightarrow модель легче
- выделение фильтра с векторной семантикой

Сверточные автокодировщики

Классические операции декомпозиции: число \rightarrow матрица kxk

Коды с разностенциальностью: gh / vdemoulin / conv_arithmetic

В полученной сети 255 весов, в классической было больше 300!

Написавшая исходную кривую не будет иметь нулей и пропусков

\rightarrow большие времена не обрезаны

\rightarrow нужно больше единиц (даже с повторением)

Свертки можно считать формой нейронизации для единиц с проприетарной структурой

Лекция 6. Рекуррентные нейронные сети, или как правильнокусать сеть за хвост

6.1 Мотивацию: обработка последовательностей

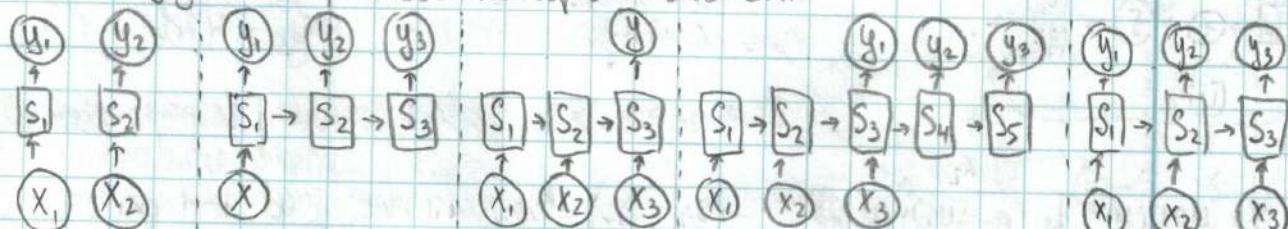
Раньше разные входы + независимо, теперь хотим последовательности
Можно читать окнами, тоже + их независимо. Но есть — свертки по времени

Чтобы “запомнить” + все предыдущие входы, есть рекурр. сети.

Сохраняет между шагами промежуточные состояния

Алгоритмика: time-delayed NN, сети с временной задержкой (memory подачи входов)

Типы задач с обработкой последовательностей:



- one-to-one
- \rightarrow состояния не пересекаются
- \rightarrow ~ одинаковая обработка

- one-to-many
- \rightarrow пересекаются
- \rightarrow zB анонсирование картины текстом

- many-to-one
- \rightarrow классиф. текстов

zB sentiment analysis

- many-to-many
- \rightarrow дополняют друг друга
- \rightarrow машинный перевод
- \rightarrow параллельные системы

- synchronized many-to-many
- \rightarrow zB разметка видео-новости

У скільки видів Качрати про RNN

- RNN походить на прогресивні з багато перенесеними. Ось Таксоніз - погляд!
- можна & статистичну задачу як посл-стю: "оскільки" картинки, наявніше розглядання
- температуре β softmax: чим вище β , тем більше "імовірності" вер-сті \Rightarrow рандомізовані, які більші ампліуд.
- чем нижче β , тем більш конфідентні, які конфідентні \Rightarrow увереніші, які консервативні
- висулюємо:
 1. прості вер-сті рандомізовані в будь-який момент
 2. залежність отриманого підтримки в пам'яті: прост. наявність b [1], пам'ять блокноти, кінець строкі, внутрішні члені, if-членів etc
- приклад з порівнянням корреляції хмл, потрібно компар. later, синхрон. Comp. C!

6.2 Розширені моделі та архітектура RNN

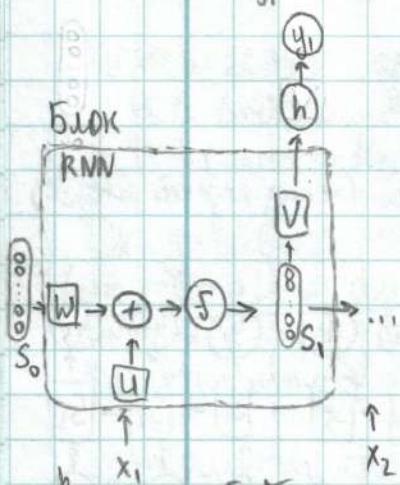
$$y_6 = f(x_3, x_4, x_5, s_2) = f(x_3, x_4, x_5, h(x_2, x_3, x_4, s_1)) =$$

наявніше состояння

$$= f(x_3, x_4, x_5, h(h(x_2, x_3, x_4, h(x_1, x_2, x_3, s_0))))$$

Получаємо, \rightarrow обчислюємо на t шарі реєт з урахуванням попередніх
 \rightarrow градієнт проростає через кілька блоків залежності

Архітектура "простий" RNN:



W - матриця весів для переходу лемту експертами состояннями
 U - матриця весів для входів, V - для виходів
 f - підмінність рекурр. функцій (sigmoid, tanh, ReLU)
 h - функція яка повертає обсяг (зб SoftMax)

$$a_t = b + Ws_{t-1} + Ux_t \quad s_t = f(a_t)$$

$$o_t = C + Vs_t \quad y_t = h(o_t)$$

Слой RNN - це блок, рефериуючи на будь-яку посл-стю

В момент T функція активізації $L(0, \dots, o_T)$ виконує веси W + 1 раз

• shared weights: • зручно хранити - будь-які ваги матриця що існує
 • градієнти не дуже залежать до цих ваг, як в широких слоях

Проблемы RNN:

1. вырождающиеся градиенты (если при 1 проходе норма вектора сильно растет, а тут T)
2. блокирование входа \rightarrow то есть уединение - проблема с длинными цепями
3. норма с норм. более сложных, кирличиков" настройки решения

Но это всё однозначно! Это можно сделать чтобы избавить от частей архитектуры на многослойное.

? Решение выходов: (?! какой выход хотят? как выбирать, анализировать?)

1. Функция от входа к скрытому слову. Инициализация: "надает" временнюю структуру памяти посред. шлагами, и склонные обогащают признаки дальше уже обрабатывая чем-то простым. Пример: распознавание речи
 2. Функция от скрытого слова на выход. Инициализация: нужно расставить склонные факторы выделенного представления. Не обдувательно заменять глубокой сетью, можно чем-нибудь напоминающим
 3. Функция перехода между склонами. Но склонные временные цепи то, что это шаг не могут распространять по времени. Но есть и решения - shortcut соединения
 4. Новейшее: + есть RNN как шаг в глубокой сети, или её выходов так выходов след. слов. Тогда есть учит свой "масштаб времени" (как раньше скрипты)
- ? ? не очень понимаю архитектуру?

Направленные сети: решают проблему забывания начальных состояний в момент, пока добегут до конца. Идея: запускаем RNN с двух концов неравномерно, вспомни, что получаем на t элемент не для состояния и глубине выходов при обработке, получая в результате один ответ на выход, склоняясь слева и справа!

На практике bidirectional LSTM или GRU

Часто RNN используют не для того там, где склонны быть не прямые и эффективные

6.3 LSTM (Long Short-Term Memory)

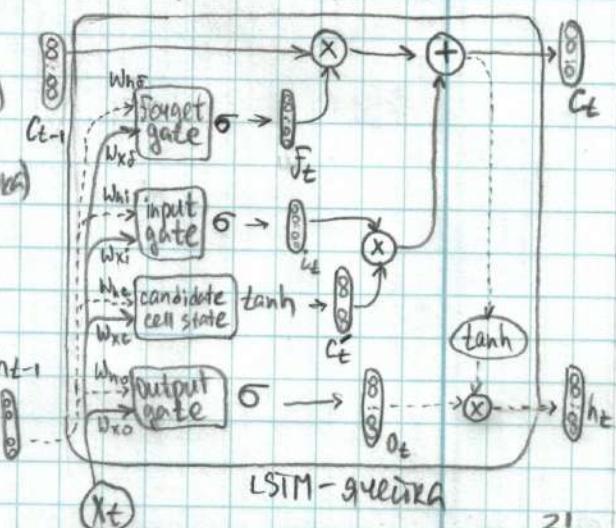
h_t - вектор скрытого состояния в момент t (выход блока)
 C_t - состояние склонки памяти

Forget предыдущий input x_{t-1}

$$C_t = F_t \odot C_{t-1} + I_t \odot C'_t$$

Помимо этого умножение \rightarrow можно добавлять членами

Благодаря σ есть множества и множества)



Те же отмеченные, но более точнее формулировки долгосрочные зависимости?

- предположим, $f=1$, т.е. ничего не забывает \Rightarrow наметь $c_t = c_{t-1} + i_t \odot c'_t$
- $\Rightarrow \frac{c_t}{c_{t-1}} = 1$ и при backprop можно не менять наметы, пока не уходит
- также это решает проблему запутывающихся градиентов
- NB: вдруг неправильные. Долгосрочными числами, а не $N(0, \sigma^2)$, тогда $f=1$ в начале

Распространенная модификация (с 2000 года, это, не мало для этого)

В управлении c_t пред. наметы присутствуют только через $h_{t-1} = o_{t-1} \odot \tanh(c_{t-1})$

т.е. если не даем значений на выход, то и на пред. наметы выйти не можем! Ужасо!

Решение: добавляем рефлёны: вход c_{t-1} в гейты i_t, f_t, o_t (с собственными весами)

Есть множество разных вариаций: убрать гейты, ф-ции активации, заменить схемами, $\neq i_t = 1 - f_t$ - своеобразные гейты, долгие рекурр. связи между блоками...

Но исп. [336] показало, что "базовый" LSTM решает.

- Ключевыми комп. задачами являются забывающие гейты и ф-ции активации не бинарные
- Удаление "замочного механизма" почти не ухудшило метрики
- Сокращение f и i тоже не особо ухудшило! хотя сделано сильно проще

6.4 GRU и другие варианты

У LSTM много блоков, которые к тому же "разбрасываются" по всем ресурсам
но не все они нужны. Важнее всего output, forget и намет наметь с

GRU (gated recurrent unit): включаем входной и забывающий, также h и c

$$u_t = \sigma(W_{xu}x_t + W_{hu}h_{t-1} + b_u) \quad - \text{гейт обновления (update, } O + f\text{)}$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \quad - \text{гейт передачи (} h + c, \text{ что переключит блоки)}$$

$$h'_t = \tanh(W_{xh}x_t + W_{hh}(u_t \odot h_{t-1})) \quad - \text{кандидат на след. выход}$$

$$h_t = (1 - u_t) \odot h'_t + u_t \odot h_{t-1} \quad - \text{след. выход}$$

? с. 251 - интересно?!

r_t - как обновляющий входной блок с изменяющейся наметой

u_t - какую часть оставить неизмененной

Мемори h_t и h_{t-1} мин. связь \Rightarrow карусель конст. ошибки \Rightarrow прод. но залуженог

Основное отличие GRU от LSTM - есть f размежуливающий h и c . Так же нет второго неизменяющего на выходе блока. Работает так же хорошо, а временные

Варианты где напр. блоков с одинаковыми частотами - Phase LSTM, если одновременно есть времена, то напр. то шир. синхронного прогенератора)

22. В Google переделали очень автономную архитектуру, есть и получше GRU

6.5 SCRNN и другие: долгое хранение в обычных RNN

Одн. проблема RNN - затухание (но не всегда экспоненциально). В LSTM и GRU модель забывает ее явно, но уменьшает модель, тут говорим о других проблемах

Проблемы с пропусканиям (затухание и перенос) не обусловлено не столько самим арх. структурой, сколько структурой сети и начальным преобразованием
Сеть Элмана: нейроны конекции получают на вход собственные значения c_{t-1} и выхода нейронов прошлого слоя (один к одному)! Р. без нелинейности!

SCRNN (структурно-континуированный RNN, структурно-пропускающий и.с.)

Два разных скрытых слоя: одинный се с полносвязанным весами и неприменяющим конекциями C_t с единич. н. весом

$$C_t = (1-\alpha) A x_t + \alpha C_{t-1}$$

$$S_t = f(P C_t + U x_t + W S_{t-1})$$

$$y_t = h(V S_t + B C_t)$$

C_t и S_t можно смотреть как один слой с разной структурой весов
то сп. с LSTM так же (при $\beta_1 = 0$), модель лучше (если много единиц)

Другой подход - н. весов между скрытыми состояниями с един. нормой - разрешают но сп. не предъявляют и не засуждают. Чистой путь - иниц. единичной, тогда при Kelly векторе весов все ок (также как повторение его улучшается почти до уровня LSTM)

Следнее: разделяющееся на блоки произведение на S_t и на S_{t-1}
или неизвестные параметры умножающие матрицы (unitary RNN, uRNN)

Это постепенно улучшает ситуацию в среде, но универсальной "переменной" пока нет, это сильно ограничивает

6.6 Пример: восстановление текста

Как читать текст для обучения? в конечном итоге

→ основ - просто, но случайные отрывки из текста будут мешать

→ предложенные + специальные за конца - будут завершающие результат

→ длинные тексты читать на недельные, шесть. скротые состояния предваряющим текстом линии-батом. Круто, но требует технической работы

Четвертый - sequences - ТЧЧС - выделяет место на полу текста, а не в конце

clipnorm / clipvalue - ограничение градиента, чтобы не бурчались

"искусственное же значение" ограничение изначально

? Skip-layer connection (связь с пропуском слоя) - связь след. слоя по тому же выходу предыдущего, но и его пред. - быстрые уроки для последних слоев
Можно сделать быструю концептуальную обработка

Часть III. Когнитивные архитектуры и применение

Глава 7. Как научить компьютер читать

7.1 Императивная обработка текста

Хомски: корпоративные грамматики, phrase-structure based parsing

Много ограничений, но в реальности нужно много контекста и "здравого смысла"

Пример: предложение ANDROID (к чему он. значение) commonsense reasoning

Что такое "понимание текста"? Важно много задач:

1. "Человеческие": хорошо определенные задачи, знаюе, что же там пишено
Примеры: синтаксис, лингвистика, часы речи, фразео-предиктивный лингвистический анализ
2. Для которых нужно понимание текста, но известные задачи не еще просты
Примеры: языковые модели (транс. символ), IR, точность, векторные пространства
3. Несколько текст + некоторые метрики, которые уже не всегда очевидны
Примеры: корпоративные abstract, машинный перевод, качественные модели

"При мерами для измерения широкого понимания BLEU (Bilingual Evaluation Understudy) и подобных. Идея в модификации precision для сравнения отдельных моделей и разных языков аналогично, т.е. с π по n -граммам. Но есть проблемы:

1. Не применять пред. тексты, т.к. диктор, значим
2. [23]: корреляция этих метрик с человеческими оценками - около 0!

пример: м/у метрики и реальн. 35%, т.е. предварительная оценка есть, но неко неформальная

Еще одна задача, где вход - не символы. Например, для распознавания речи очень важно иметь представление о звуке (или могут многое убыли добавлять)

7.2 Распределенные представления word2vec

Distributed word representations aka word embeddings $\in \mathbb{R}^d$, для которых базовое предположение - если вхождение в R^d будет соотноситься семантическим образом

В машинном обучении есть дискретивная проблема — похожие слова берутся из разных контекстов. Это приводит к переходу от one-hot encoding входа модели ОНЕ к векторам слогонумерам: • частичное представление • не учитывается зависимость между похожими словами (корней, синонимов)

Для такой модели не нужно разделять текста (из русских что говорят темат. корпуса)

Соединение матриц: в строках слова, в столбцах "текста". Он большой, содержит до $M \times N$ слов и $N \times K$ векторов признаков как $X \approx U V^T$ ИММ(N, K) ИММ(M, K) где матрица K . Тогда U — векторы признаков для всех N слов

Это явно латентного семантического анализа (LSA / LSI) $\begin{bmatrix} M \\ N \end{bmatrix} \approx \begin{bmatrix} M \\ N \end{bmatrix} \begin{bmatrix} N \\ K \end{bmatrix}^T$

Есть проблема со слишком часто встречающимися словами (стопы), предлоги, есть разные подгруппы

Тематическое моделирование (topic modeling): происходит из базового LSI вероятностная интерпретация. Текст — вероятное распред. тем, тема — вероятное распред. слов.

$$P(w|d) = \sum_t P(z_w=t|d) P(w|z_w=t)$$

вероятн. слова в тексте тема слова

Задача: обучить параметры распределений по частотам встречаемости

Этот метод очень распространён и много рассматривается. Часто его можно встретить в интерпретации. Но все еще не имеет своего языка моделей

2008. работы Бенгрина по направлению языковых моделей

- У слова сопоставлен текст-embedding
- выражим вероятн. появление слова в контексте как сумма от векторов признаков
- подобрали f и эмбеддинги, максимизируя правдоподобие корпуса текстов
- в КЧ-ке можно было сеть

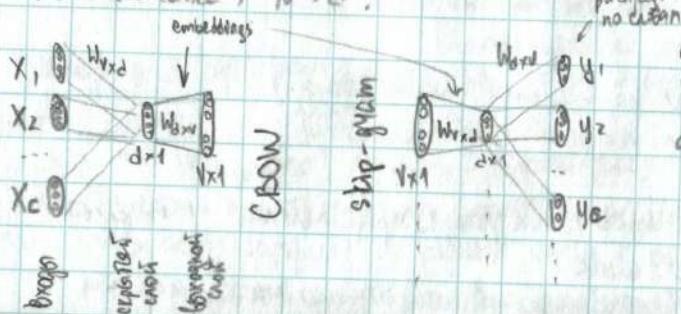
word2vec: учились представлять слова из контекста и изолиров

→ continuous bag of words: представляют слово, усреднив контекст

- один скользящий окно. В нем группа из N строк — представление N слов слова
- простое правило, чтобы полученные полученные векторы тоже были корректны
- обучение модели с целью сделать априорное распределение похожим на реальное

→ skip-gram: но дальному слову представляется все слова контекста

? Рис. 7.2 — опечатка? $N \rightarrow d$?



C — размер контекста
V — размер слова
d — предлагаемый размер эмбеддинга

Чем же тогда? Оно из односвязей - то слово просто добавляется к конфигурации самого себя, есть якобы запятая 0, а это никак наводит на раз-р

Оптимизирующая некая сумма, предпол., что получится удобный нам раз-р.

Примитивное синтаксическое: где в себе отображены некоторые структурные особи, примеров (авто не в этом конфигур.) и часть суммы чисел только из них, соотв. добавляем на уме не все бэс, а только у небольшой части слов \Rightarrow объект

7.3 Русскоязычный word2vec на практике

B термины

7.4, Glove: предсказывает напарнику правильного

Надо что-то спредеить между н. корректности (кд. не знает про название слова) и word2vec (коф. значение н. корр.).

Моделируем из н. корректности отношений вероятностей P_{jk} - где одно слово j где

$$\text{Считаем } P_{ijk} = \frac{P_{ij}}{P_{ik}} = \frac{P(j|i)}{P(k|i)} = \left[\begin{array}{l} \text{б. вероятн.} \\ \text{б. ожиданн.} \end{array} \right] = \frac{P_{ki}}{P_{kj}} = \frac{P(i|k)}{P(j|k)}$$

где w_i, w_j - первое слово, w_k - второе конекстное слово k .

Далее предположим F из множества и с отр. вб-баки \Rightarrow экспонента

Обдумаем с учетом пар, включающим на конекст. парам слов

Что Glove - один из двух "стандартных" методов пары с word2vec

Чт. еще про двойственные распределения, интересно про аномальность

7.5. Вектор и вектор от предложений и тд

Как обобщить от слова к предл. и текстам?

1. Несколько комбинировать вектора слов. Скорее беспорядок
 2. К векторам слов в word2vec добавляется вектор аддона
 3. Частотами на уровне фразе skip-gram \rightarrow skip-thought
 4. Encoder-decoder для предложений, считает по времени (время послр.)
- /* хороший пример с sentiment analysis с послр. словами */

Проблемы:

- Не уч. учение об однозначности, их распред. представлений не совпадают
- Сложно смысльное значение слова
- Как-то слово в предложении лучше нумеровать

Это приводит нас к идее носижевольных представлений (character-level representations).
Хорошо это или нет, но "Маррено" неподобающе в аудиоизомах не звучит..."
Есть носижевольные работающие модели на основе LSTM или сверточек. Учитом что экономии
времени можно предъявлять слова чётко представлений заранее

Чубокие структурированные лингвистические модели (DSSM): используют векторное
значение слов: слово \rightarrow [стол, машина, лоб, оба, вон]. Слово векторизует поборь (AI³, A-аффинат)
Также лучше с лучшей обработке определите

Самое простое разбиение или обработка на уровне слов - FastText. Или же текст. предс.
грамматик, используя их, чтобы получить вектора слов, которых состоит строка
или skip-грамм.

7.6 Рекурсивные нейронные сети и синтаксический разбор

Что если в тексте не как построить рекурсивный синтакс. разбор?

Строим рекурсивную сеть так: $X_{t+1} = F(W_L X_{t+1} + W_R X_{t+1} + b)$

X_t -текст. предс. ближайшего t , F -нейрон. ф-ция активации W не меняется и не удаляется

- но если - рекурс. сеть с очень особыми свойствами, такие обучаются вспомог.
- можно сделать инверсию

Но для этого нужно изобрести деревья разбора. И у них есть большие проблемы,
например, с не-простивильными деревьями, или же неоднозначностями, когда модели решают
засчет знаков между словами

Часть 8. Современные архитектуры, или как в споре рождается истин

8.1 Модели с вниманием и encoder-decoder

Внимание в смысле фокусации информации

Классическое CN: от первого всех окон к каскадным моделям (сегментный классификатор
тогда же, но что интересно) или к более сложн. "интересности" областей (т.е. по переносу контекста)
Многие годы сети контролируют контекст, путем внимания

Пример для картинок: рекурс. сеть, которая не знает о том, что может настичь след. потому бывает,
бывает нет. "смотрит" с разной внимательностью

Почему это то "бес" чисто изобретение

- hard attention: но этим весам дословно художник (но градиенты не прокачиваются)
- soft attention: смотрят на разные части с разными весами

* в 10.3 будут формализованы промежуточные

- Encoder-decoder: сбрасывает вход в репрезентацию, потом её разворачивает
- в что-то иначе (перевод, описание картинки etc)
 - у этой архитектуры есть проблема с длинными пред., когдा картинка содержит много текста.
 - также есть разные виды. Это можно решить вниманием
 - soft alignment: есть веса, как слова входа влияют на текущий этап перевода
 - soft attention: по картинке состоянию и сконденсированному входу будет переводиться

Hu (2016) считает что переводчик состоит из кодековщика, декод. и сети внимания

- множества ресурс. слов (но не только в EN и DE)
- при этом что-то типа residual связей для обмена между блоками
- 双向 направлениий внимание (контекст слева и справа)
- удаление слов из фразы

ED помогает в малых языках, особенно если Е двухуровневый. текущая фраза + контекст (ресурс) - HRED архитектура

8.2. Генеративные модели и глубокое обучение

- дискретные генеративные модели описывают $f: x \rightarrow y$. Вероятность: частота условного расп. $p(y|x)$
- генеративные - частота совместного распределения $p(x,y)$ NB: $p(y|x) = \frac{p(x,y)}{p(x)}$ $\propto p(x,y)$
- 1. критика реальной, 2. не обозательной - может учить прошлое $p(x)$
- 2. отн. правдоподобие или расстояние между распределениями (кудаика - лейблера)
- 2. - очень неравнозначно: ясно как много известных или новых - только из пред. нородительских еще надо учить. На слайд есть схема вариаций моделей (с. 345)

8.3 Генеративные сети

Генеративные модели без учёта обучения становятся

Generative Adversarial Networks (GAN) - генеративные состязательные сети

Генератор, дискриминатор

$$G = G(z, \theta_G): Z \rightarrow X \quad Z - \text{нр-бо} \text{ серийных (гифтовых) фрагментов расп. } p_Z(z) \quad (\text{ист. норм.})$$

$$D = D(x, \theta_D): X \rightarrow [0, 1] \quad \text{Бывает бесп. что критик поглощает}$$

$$\underset{G}{\min} \underset{D}{\max} \left(\mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p_Z(z)} [\log (1 - D(G(z)))] \right)$$

Алгоритм обучения: на t шаге D^t генератор вырабатывает все одно и то же, поэтому все должно быть док-бо, что это шаги и приводят к спиральности расп. к p_{data}
 то шаг D критика на t шаге добывает до спиральности, т.к. задана суммарная, но в реальности просто делают несколько шагов

8.4 Пространственный генератор и дискриминатор с дополнительной информацией

Равнодействующее расп. \rightarrow нормальное

Вход G многочлен, где от 0-ых активаций

D где многое G!

Пример не работает! Текст не имеет интереса!

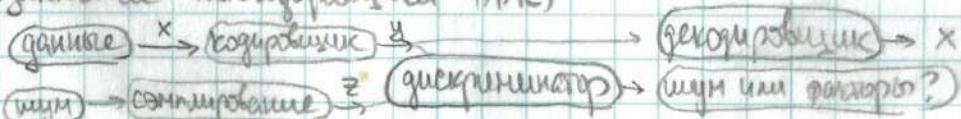
8.5 Архитектуры, основанные на GAN

Почему хорошо работает? Рассуждение про $KL(P_{data} \parallel P_{gen})$ и $KL(P_{gen} \parallel P_{data})$
В 1 случае в мультифакторном расп. получаете эти «пересеченные», а в втором - одну из них
Описание нескольких видов

- исходный GAN
- DCGAN - deep convolutional G и D. Есть общий ядро

Есть красные приложения, которые с разн. зависимостями. Super-resolution
- Conditional GAN - на вход подается метка класса y (и G, и D)

✓ ЗБ напоминает мне научную выставку
- InfoGAN - «распутанное» представление (disentangled repr.) - с антеград. приложением
- StackGAN - пример перевода текста в картинку
- против склоняющихся, но может считать loss G по D из будущего (хотя S неизвестны)
- состязательные автогенеративники (AAE)



дискриминатор создается с кодером, у становится из расп. $P_2 \Rightarrow$
потом для генерации можно будет сгенерировать из P_2 .
красиво!