

Избранные изображения

Часть 1. Как открыть изображения сети

Задача 1. От Титаника к информатике (6-38)

1.1 Революция

или we need to go deeper

1994: изображения сети - это хороший способ сделать практический шаг

Хипотеза, середина 2000х: преобразование Титаника сейчас хорошо и без него, это даже интересней

Второй разброс: большие физич. множеств (2006 - идея вен. GPU)
множество 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 началось через Чемпионат: М, М, первый, М прибл. М, М
помогает себе разобраться. Компьютер не лучше и.
Чтобы быть лучше нужно было где-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 методом (определить распределение)
 - Снижение разницы (так, что можно \neq баск. исходные)
 - Кластеризация
- Обучение с частичной привлечением учителя
(использовано, если разница слишком, а неравнозначно)
- Обучение с учителем
 - Кластеризация
 - Регрессия
 - Обучение различиями
- Обучение с подкреплением (напр. при решении задач A/B) ??

1.4. Особенности человеческого мозга

Мозг работает параллельно в зависимости от задач согласованности нет синхронизации (когда же, например, зевет дверь в спальне)

Распознавание лиц делает ученые за <10 мсек.

т.е. мозг очень параллелен и имеет добавлено мессенджеры - GPU, где CPU

есть много "горизонтальных" связей, связей между нейронами одного и т.д.

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

Человеческий мозг - есть специализированное устройство, но нейроны не всегда передающие (ранние боли!), обрабатывающие нейроны и синапсы

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

1.5. Презентация вопросами: что это не самое лучшее решение?

"Может ли нейронная сеть наука?" *Митч Калли
Процессор MOS 6502 3510 транзисторов, 1.5 ГГц за секунду вычисляет
много задач, которые невозможно решить вручную

1.6. Базис и модели современных ИИ

Многие хороших примеров из-за требований человека (распознавание лиц, игр). В чем отличие от концепции Тьюринга, например?
Во-первых человека можно было бы обучить: числа, мн-ва, новичок
не замечает частей.

1. Интуитивная рука: крайне приближенные фиг. модель мира, способная к очень мощным обобщениям и переходу на новые виды выхода
2. Интуитивная психика. Например, обучение Марии членам ее семьи, что так и неизвестно

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

Можно породить абстракции из очень небольшого числа примеров
типа тех же через one-shot learning

в-третьих, причинность (causality) - способность воспользоваться, используя
"причины" происходящего. Например, научить то роботу раздражать
Марии, хорошие члены семьи, и улучшает эти качества

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

2.1. Частота

В моделях неизбежна неопределенность, важно ее учитывать, а
именно нужно учить ее определять

- дискретное с. фнк., как или ск. число исходов, p в сумме 1
- напр. с. фнк., p -это распределение $F(a) = p(x < a)$, плотность p -фнк. $p(x) = \frac{dF}{dx}$, $\int p(x) dx = 1$
- 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. $p(x,y|z) = p(x,y|z) \cdot p(y|z)$

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 оценить вер-сю текущую 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 индекс max likelihood: $\theta_M = \arg \max_{\theta} p(D|\theta)$

→ в байесовском подходе и обр. ML ищут max апостериорную индексу
 $\theta_{MAP} = \arg \max_{\theta} p(\theta|D) = \arg \max_{\theta} p(D|\theta) p(\theta)$

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

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

$$p(y|D) = \int_{\theta} p(y|\theta) p(\theta|D) d\theta \underset{\substack{\text{апостериор} \\ \text{норм.}}}{\approx} \int_{\theta} p(y|\theta) p(\theta) p(D|\theta) d\theta$$

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

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

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

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

Почему RSS ? Это max likelihood в предположении о нормальном шуме

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

2.3. Рассстояние Кульбака-Лейблера и перекрестная энтропия

Ассиметричный критерий \Rightarrow град. спуск. не помогает

Рассстояние Кульбака-Лейблера (относительная энтропия) - мера нехожности
двух вероятностных распределений $P \neq Q$. 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

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

NB: 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 не лок., но не глоб. максимумы, более того, у склонов их много
Ну идем дальше, что же такое - градиентный. Эвристические методы решают!

Математическое описание: если с производной не очень, нужно минимизировать

1. приближенное вычисление производной
2. производная \rightarrow функция, приближающая исходную
3. производная \rightarrow дополнительная функция ошибок

25. Градиентный и градиент на нем

Представляем склонную функцию как композицию более простых
Элементарных и сложных функций, где глоб. максимумы ее саму и
ее производную по другому

Хотим минимизировать склонную функцию ошибок (loss)

- $f \circ g$ - склонные функции $\Rightarrow (f \circ g)'(x) = (f(g(x)))' = f'(g(x)) \cdot g'(x)$
- f -ベкторная, g -скал. $\nabla_x f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \Rightarrow \nabla_x(f \circ g) = \begin{pmatrix} \frac{\partial f \circ g}{\partial x_1} \\ \vdots \\ \frac{\partial f \circ g}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x_n} \end{pmatrix} = \frac{\partial f}{\partial g} \nabla_g g$
- обе векторные, $\Rightarrow \nabla_x f = \frac{\partial f}{\partial g}, \nabla_x g = \dots + \frac{\partial f}{\partial g_k} \nabla_x g_k = \sum_{i=1}^k \frac{\partial f}{\partial g_i} \nabla_x g_i$
то же, что мы учим: $\nabla_x f = \nabla_x g \nabla_g f$, где $\nabla_x g = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \dots & \frac{\partial g_k}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial x_n} & \dots & \frac{\partial g_k}{\partial x_n} \end{pmatrix}$ - матрица
Экспон

• мы можем композиции можно применять
6 двух направлений: • от новых к старым, получая ч.п.разн. А ЧПР и
старый и новый не переменной = выход • от старых к новым, получая
ч.п.разн. старых но всем тренирующим

• В МЛ нужны 2-й вариант - это же loss, который есть на-бэккомпьютер
• Это и есть back propagation, т.е. в тренировке, передает направления

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 обучения, просто модель
- Идея о включении испытывающихся досяг \rightarrow обучение по Хеббу (сигналам)
- Идея о вертиках (после прокачки битовых нейронов)

3.2. Как работает перцептрон

для приближения

* 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 $y(x)$ зону

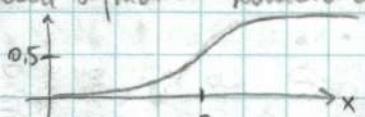
Выводим формулу ошибки: хотим минимизировать без разрывов

Формула 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, обновляем $w^{(t+1)} = w^{(t)} - \eta \nabla_w E_p(w) = w^{(t)} + \eta z_n x_n$

Когда ввести критерий активации, иначе содержит только сеть не построить. Классический вариант — логистический синтакс

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



Для monotonicity необходимо,
 $f \rightarrow 0$, $f \rightarrow 1$

?! 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 можно сортировать их по сорту и отсортировать!

Задача И.И.: начали разработание в машинном переводе, потому что переписывали изображение сколько-то групповых переводов, что отталкивало 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 $tanh$
Линейная тангенс танх	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f^2(x)$	$(-1, 1)$ тоже более гладкая, $\sigma'(0) = 1$, расчет и подобие танка σ : $tanh'(0) = 1$
Гипербол. (гипербolic) танк (tanh)	$\begin{cases} 0, & x \leq 0 \\ 1, & x \geq 0 \end{cases}$	0	0-само восходящий танк, не имеющий норм на конечном конце, но не имеет краев
ReLU rectified linear units	$\begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	$\begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	<ul style="list-style-type: none"> одно изображение функции \rightarrow сама линия стек. подчиняется (ан (1))-правилу приводит к неизучаемой норме (2)
Softplus	$\log(1+e^x)$	$\frac{1}{1+e^{-x}}$	<ul style="list-style-type: none"> исследование, сумма ∞ для дополнительно, так как 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 (здесь) здесь упрощает расчет, так как a отвечают за гладкость
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"> отрицательное значение и отрицательное значение затем наслаждение единого градуса гладкость

Несложные виды нейронных приложений в кибернетике - 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
 - бросо быве член - выв. 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 Таблицы 6 2005-2006

А.И. Иванченко, метод группового учёта аргументов. Осуждаем "каго"
 Свой \neq своем, исп. выхода. пред. как входы след.

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

Появились различные инструменты регуляризации и оптимизации (градиентный спуск, деснект, нормализация мин-макс, инициализация, различные виды нормализации)

3.6. Пример распознавания рукописных цифр на TensorFlow

MNIST - «глобальная машина обучения» - Ридерхи Хилтон

Класс 4
Быстрее, чище, сильнее
или об обратах, длинах и границах

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, борьба с overfitting, αL_2

Деснект - статистика-Монте-Карло-регуляризатор. Это в некотором смысле рандомизированная версия р, что это некоторый весовой сетью обновляется как обычно, но берутся микроР. p. 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 стабильней

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: есть ограничение размера батча, чтобы были адекватные статистики

! Следующая глава огромна, а у меня Рутин и команда работают

? Как они выбирали подборки к главам?

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: использует не только ' γ ', но и '' (= метод бывшего положения)

т.е. приближает 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

- поправочное уменьшение: $\gamma \leftarrow \gamma / N$ раз в N эпох
- уменьшить, когда прекращает уменьшаться ошибки на validate
- trade-off между выполнительным баллом 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 стабильные

? Аддитив: 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 большей 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 начавшегося
- работать на train, validate (параметр, остановка) и test (finalное качество)

Глава 5. Сверточные нейронные сети и автодифференциация, или не верь глазам своим

5.1 Зрительная кора головного мозга

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

Части зрительной коры:

- V1 - локальное признаки небольшого участка
- V2 - проектируем лок. признаки, чтобы обобщить и добавить бинокулярность
- V3 - цвет, геометрия, первые работы 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 четырехъярусную нейронную сеть. Наиболее ясно это видно на схеме непропущенных путей V2 и V5, между пропущенными. Возможно, внимание из V4 возвращается к V1 и V2. Это в МН пока никоим образом не 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 $W \in M(2d+1, 2d+1)$

$$X = \begin{pmatrix} 0 & 1 & 2 & 1 & 0 \\ 4 & 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 & 1 \\ 1 & 2 & 3 & 1 & 0 \\ 0 & 4 & 3 & 2 & 0 \end{pmatrix}$$

если следующим шагом: $y_{i,j}^e = \sum_{a,b \in d} W_{a,b} X_{i+a, j+b}$

результат свертки на уровне e \rightarrow $W_e = 4$ \rightarrow выход весов предыдущего слоя

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 1 \\ 1 & 2 & 3 \\ 0 & 4 & 3 \end{pmatrix}$$

$$\begin{pmatrix} 9 & 5 & 4 \\ 8 & 8 & 10 \\ 8 & 15 & 12 \end{pmatrix} = X * W$$

$$= 15$$

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 изображения
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 с ячейками
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)

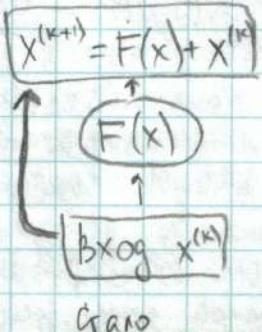
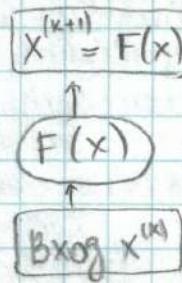
- Стартовые блоки: модули, комбинирующие сверточки 1×1 , 3×3 , 5×5 + max pooling
- Блок — объединение 4 маленьких сетей, их выходы объединяются
- Вспомогательные: 1×1 используются для нейронов, связанных для размежевания
- Число для обучения 22 параметризованых ячеек (проблема затухания градиентов):
в среднем сети включают нейроны из промежуточных признаков, чтобы выходы не могли влиять на loss (-0.03). Скорость не улучшается, но результат обучения становится 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 "входные слои" напрямую от входа слоя к выходу



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

Проблема: из-за отсутствия баланса градиентов, нет проблем с затуханием.
Можно добавлять слои, но лучше без них

БИБЛ

с весами у блоков

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 / technique / technique / веса можно удалить \rightarrow модель легче
- выделение фильтров с винтовой семантикой

Сверточные автокодировщики

Классические операции декомпозиции: число \rightarrow матрица kxk

Коды с разностенциальными: gh / vdemoulin / conv_arithmetic

В полученной сети 255 весов, в полноценной было больше 3000!

Полноценная сверточная нейронная сеть имеет гораздо больше

\rightarrow больше времени на обучение

\rightarrow нужно больше данных (даже с повторяющимся)

Свертки можно считать формой нейрорешетки для данных с пространственной структурой

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

6.1 Мотивацию: обработка последовательностей

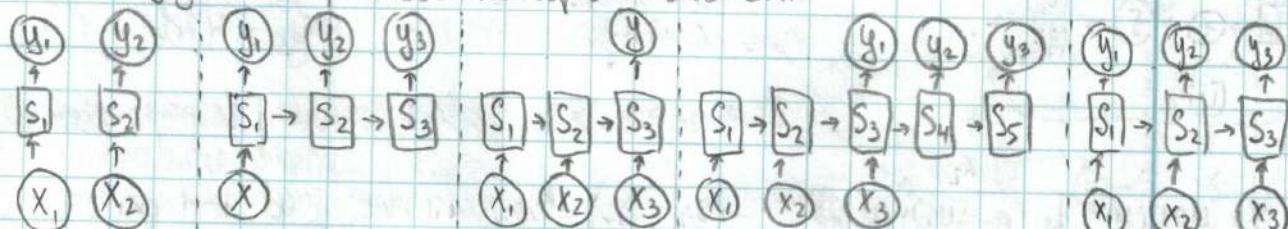
Раньше разные входы + независимо, теперь хотим последовательности
Можно читать окнами, тоже + их независимо. Но есть — свертки по времени

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 передаются
- \rightarrow ~ одинаковая обработка

- one-to-many
- \rightarrow преобразование
- блог + новостях
- \rightarrow zB аннотирование
- картинок текстом

- many-to-one

- \rightarrow классиф. новостей
- \rightarrow определяет блог + спортивное
- zB, sentiment analysis
- составление, преобразование + блог
- \rightarrow машинный перевод
- \rightarrow генеративные модели

- synchronized
- many-to-many
- \rightarrow zB разметка
- блогов-новосток

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 $b[t]$,
пояснення будущі, коли строки, внутрішні члені, if-условія etc
- приклад з порівнянням корреляції хмл, потрібно компар. later, синхрон. corr. C!

6.2 Розширені моделі та архітектура RNN

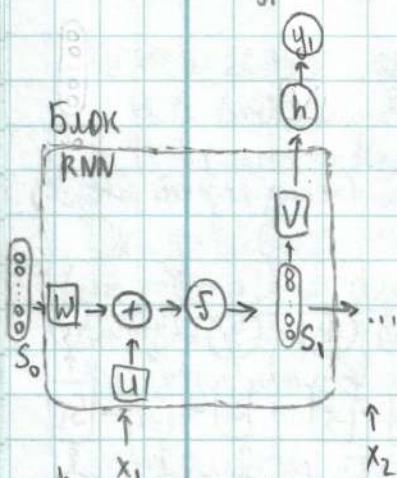
$$y_6 = f(x_3, x_4, x_5, s_2) = f(x_3, x_4, x_5, h(x_2, x_3, x_4, s_1)) =$$

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 інтервалів друк в зоні сесії

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 (зок 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, рефериуючий на будь-яку посл-стю

В момент T функція активізації $L(o_0, \dots, o_T)$ виконує веси W + 1 раз

• shared weights: • зручно хранити - будь-які відхи матриця що кути.
 • градієнти не дуже залежать до кути зразу же, як в широких
 сесіях

Проблемы RNN:

1. Высокочастотные градиенты (если при t шаге норма вектора сильно растет, а тут T)
2. Быстрое уменьшение входа \rightarrow но неудачи - проблема с длинными цепями
3. Норма с нач. более сложных "кирничков" настройки решения

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

? Решение подходит: (?! какой подход можно? как выделять, анализировать?)

1. Функция от входа к скрытому слову. Инициализация: "помимо" временному структуре памяти посред. шагами, и склонные обогащают признаки дальше. Чрез обработками чем-то простым. Пример: распознавание речи
 2. Функция от скрытого слова на выход. Инициализация: нужно расставить склонные факторы выделенных представлений. Не обдувательно заменять глубокой сетью, можно чем-нибудь напоминающим
 3. Функция перехода между склонами. Но склонные временные цепи то, что это шаг не могут распространяться во времени. Но есть и решения - shortcut соединения
 4. Новейшее: + есть RNN как шаг в глубокой сети, иен её выходят как выходы слов. Тогда есть учит свой "масштаб времени" (как раньше склонки)
- ? ? не очень понимаю архитектуру?

Направленные сети: решают проблему забывания начальных состояний в момент, пока добегут до конца. Идея: запускаем RNN с двух концов неравномерно, с разной весами. Получаем на t элемент по два состояния и двумя выходами предобработанными, их обрабатывают, получая в итоге один ответ на выход. склоняется сева и спрашивает!

На практике bidirectional LSTM или GRU

Часто RNN используют не для слов там, где склонки тоже это кроме и эффективнее

6.3 LSTM (Long Short-Term Memory)

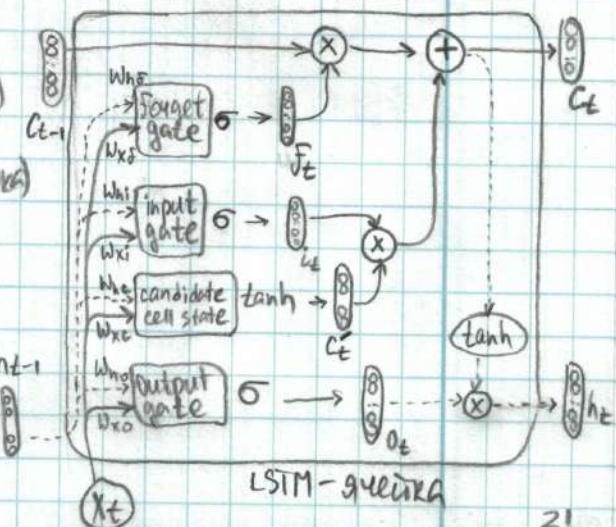
h_t - вектор скрытого состояния в момент t (выход блока)
 C_t - состояние склонки памяти

Forget предыдущий input x_{t-1}

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

Помимо склонок \Rightarrow можно добавлять числом

Благодаря σ есть множества и множества)



То есть, необходимо отдать долгосрочную зависимость?

- предположим, $\tilde{t}=1$, т.е. ничего не забывает \Rightarrow пометь $c_t = c_{t-1} + i_t \odot c'_t$
- $\Rightarrow \frac{\partial c_t}{\partial c_{t-1}} = 1$ и при backprop можно не менять помету, пока не уходит
- Такие это решает проблему запутывающихся градиентов
- NB: вдруг иначе. большие числа, а не $N(0, \sigma^2)$, тогда $\tilde{t}=1$ в начале

Распространенная модификация (с 2000 года, то, не помню где было)

В управлении c_t пред. помета присутствует только через $h_{t-1} = o_{t-1} \odot \tanh(c_{t-1})$
т.е. если не дает значений на выход, то и на помету выйти не может! Ужас!

Решение: добавляем peepholes: вход c_{t-1} в ячейку i_t , f_t , o_t (с софт. весами)

Есть множество разных варианций: убрать ячейку, ф-ции активации, заменить скрытые, $i_t = 1 - f_t$ - свободные ячейки, большие рекурр. связей и т.д. ячейки...

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

- Ключевыми комп. являются забывающий ячейк и ф-ции активации не боящимся ячеек
- Удаление "запоминающих ячеек" почти не ухудшило метрику
- сокращение \tilde{t} и i тоже не особо ухудшило! хотя сделано сильно проще

6.4 GRU и другие варианты