

Порождающие модели в глубоком обучении

Сергей Николенко

Академия MADE – Mail.Ru

18 ноября 2020 г.

Random facts:

- 18 ноября в Латвии – День независимости, а в России – День рождения Деда Мороза
- 18 ноября 1626 г. папа Урбан VIII освятил собор Святого Петра в Риме
- 18 ноября 1723 г. был пущен в действие металлургический завод-крепость на реке Исети, названный в честь императрицы Екатеринбургом
- 18 ноября 1842 г. был издан указ о построении первого постоянного моста через Неву, Благовещенского
- 18 ноября 1870 г. между Туром и Парижем открылась первая официальная линия голубиной почты
- 18 ноября 1928 г. в Нью-Йорке прошла премьера первого звукового мультфильма «Пароходик Вилли» с Микки Маусом в главной роли

Порождающие и дискриминативные модели в машинном обучении

Порождающие модели

- Итак, мы уже видели глубокое обучение и с учителем, и без.
- Но можем ли мы породить что-то новое?
- Модели бывают дискриминирующие (discriminative), которые моделируют $p(y | x)$:



Dog: 96%

Cat: 29%

Duck: 2%

Bird: 0%



Dog: 36%

Cat: 94%

Duck: 2%

Bird: 1%

Порождающие модели

- А бывают порождающие (generative), которые моделируют $p(x, y)$, и из них тогда можно сэмплировать:

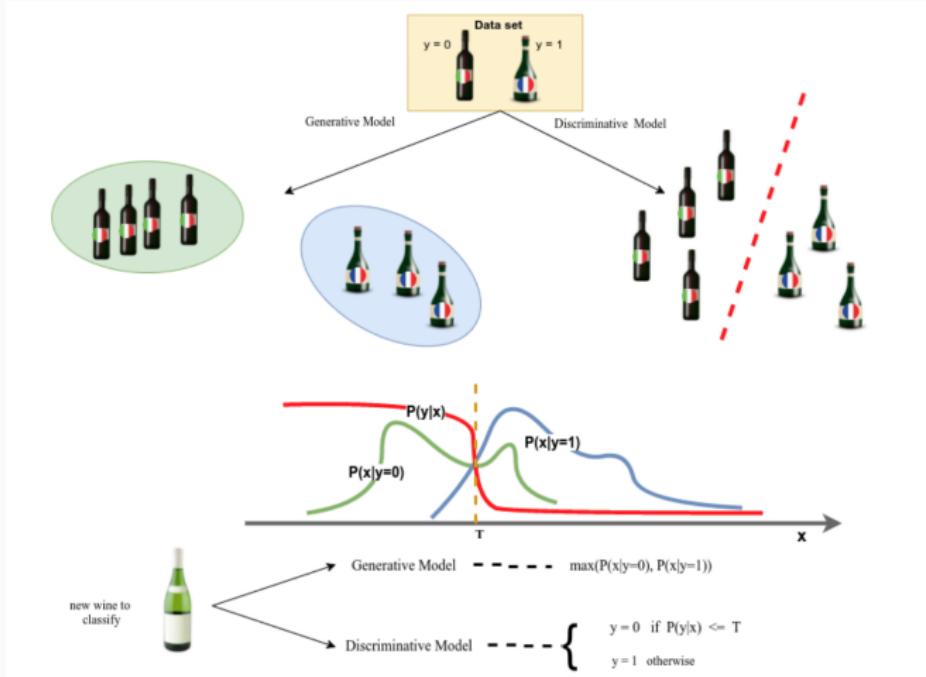
Случайный шум



- Давайте чуть подробнее. Во-первых, зачем это надо?
 - $p(x, y) = p(y | x)p(x)$, конечно, но разница есть.

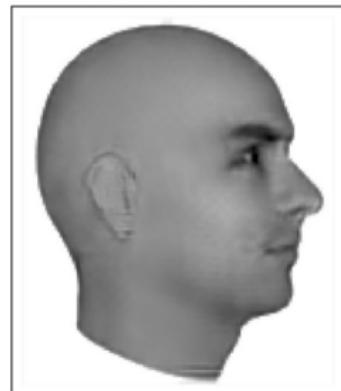
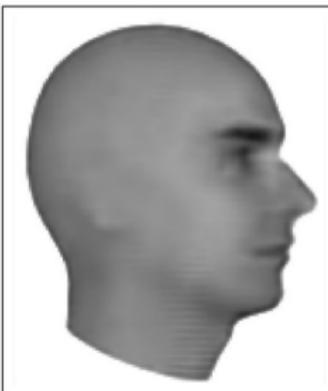
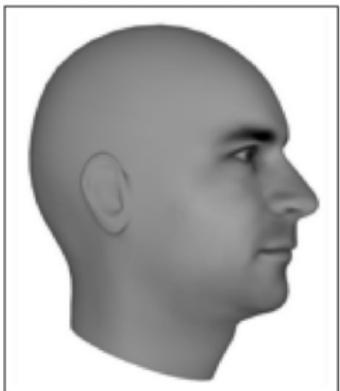
Порождающие модели

- Иллюстрация:



Порождающие модели

- Порождающие модели:
 - проверяют, насколько мы поняли распределение;
 - могут обучаться с недостатком данных и без разметки – semi-supervised learning (очень важно!);
 - могут обучаться мультимодальным выходам, когда есть несколько правильных ответов (см. ниже);
 - могут служить моделями окружающего мира в обучении с подкреплением (об этом позже);
 - ну и просто иногда действительно нужно именно порождать.



Порождающие модели

- Обычно порождающие модели максимизируют правдоподобие:

$$\theta^* = \arg \max \prod_{x \in \mathcal{D}} p_{\text{model}}(x; \theta) = \arg \max \sum_{x \in \mathcal{D}} \log p_{\text{model}}(x; \theta).$$

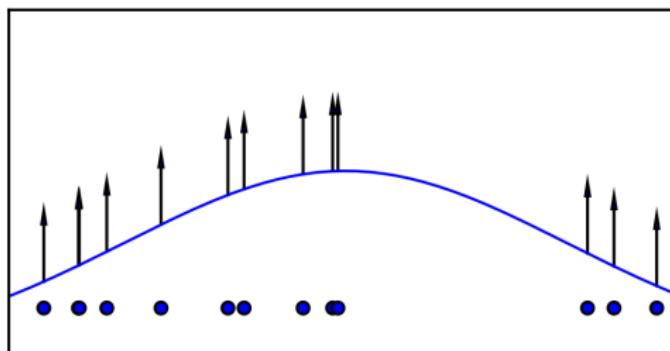
- Важный другой взгляд – это то же самое, что минимизировать KL между «распределением данных» p_{data} и p_{model} :

$$\theta^* = \arg \min \text{KL}(p_{\text{data}} \| p_{\text{model}}) = \arg \min - \int p_{\text{data}}(x) \ln \frac{p_{\text{model}}(x)}{p_{\text{data}}(x)} dx,$$

потому что p_{data} для нас дано в ощущениях через \mathcal{D} , и это всё равно что дискретное равномерное распределение.

Порождающие модели

- Иначе говоря, точки данных «тянут» вверх распределение p_{model} :



$$\theta^* = \arg \max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(x | \theta)$$

- Только в большой размерности и не просто с гауссианами всё куда сложнее...
- Большая разница в том, предполагаем ли мы, что можем явно определить и посчитать плотность p_{model} .

Порождающие и дискриминативные модели

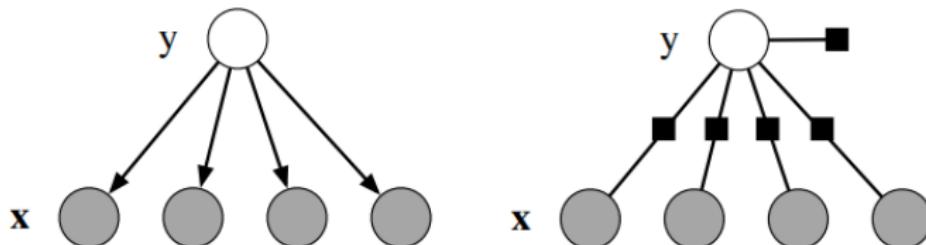
- Давайте рассмотрим пару примеров, которые дают понимание разницы между порождающими и дискриминативными моделями.
- Заодно расскажу вам, что такое CRF, всё польза. :)
- Неожиданный заход: в чём разница между наивным Байесом и логистической регрессией?..

Порождающие и дискриминативные модели

- Наивный Байес – это порождающая модель:

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^K p(x_k | y).$$

- Предположение в том, что признаки независимы.



Порождающие и дискриминативные модели

- В логистической регрессии предположение в том, что $\log p(y | x)$ – это линейная функция от x :

$$p(y | x) = \frac{1}{Z(x)} e^{\theta_y + \sum_{k=1}^K \theta_{y,k} x_k},$$

где θ – веса, θ_y – свободный член (похож на $p(y)$), $Z(x)$ – нормировочная константа.

- Можно, кстати, переписать в чуть более общем виде через признаки:

$$p(y | x) = \frac{1}{Z(x)} e^{\sum_{k=1}^K \theta_k f_k(y, x)},$$

где $f_{k,j}(y, x) = \mathbb{I}[k = j] x_j$ для векторов весов признаков и $f_k(y, x) = \mathbb{I}[k = y]$ для bias weights.

- Но из наивного Байеса тоже можно получить $p(y | x)$, ведь $p(y, x) = p(x)p(y | x) \dots$

Порождающие и дискриминативные модели

- Более того, действительно получится то же самое. У них одно и то же пространство гипотез, и одно можно перевести в другое:
 - если обучать наивного Байеса максимизировать условное правдоподобие, получится логистическая регрессия;
 - а если интерпретировать логистическую регрессию как порождающую модель с

$$p(y, \mathbf{x}) = \frac{\exp\left(\sum_{k=1}^K \theta_k f_k(y, \mathbf{x})\right)}{\sum_{y', \mathbf{x}'} \exp\left(\sum_{k=1}^K \theta_k f_k(y', \mathbf{x}')\right)},$$

то получится тот же классификатор, что из наивного Байеса.

- Наивный Байес и логистическая регрессия образуют generative-discriminative pair.

Порождающие и дискриминативные модели

- Т.е. единственная разница в том, что наивный Байес – порождающая модель, а логистическая регрессия – дискриминативная.
- Если бы мы могли получить идеальную модель $p^*(y, x) = p^*(y)p^*(x | y)$, то не было бы никакой разницы.
- Но на практике есть разница, оценивать ли $p(y)p(x | y)$, а затем вычислять из этого $p(y | x)$, или сразу напрямую оценивать $p(y | x)$.
- Порождающие модели могут больше, но у дискриминативных больше свободы.

Порождающие и дискриминативные модели

- Например, пусть есть порождающая модель с параметрами θ :

$$p_g(y, x; \theta) = p_g(y; \theta)p_g(x | y; \theta).$$

- Можно её переписать как $p_g(y, x; \theta) = p_g(x; \theta)p_g(y | x; \theta)$, где $p_g(x; \theta) = \sum_y p_g(y, x; \theta)$, $p_g(y | x; \theta) = p_g(y, x; \theta)/p_g(x; \theta)$.
- А соответствующая ей дискриминативная модель должна иметь априорное распределение $p(x)$, которое может получиться из неё, т.е. $p(x) = p_c(x; \theta') = \sum_y p_g(y, x; \theta')$, и условное распределение, которое может получаться как $p_c(y | x; \theta) = p_g(y, x; \theta)/p_g(x; \theta)$:

$$p_c(y, x) = p_c(x; \theta')p_c(y | x; \theta).$$

- Разница в том, что теперь не обязательно $\theta = \theta'$, и дискриминативная модель более выразительна.

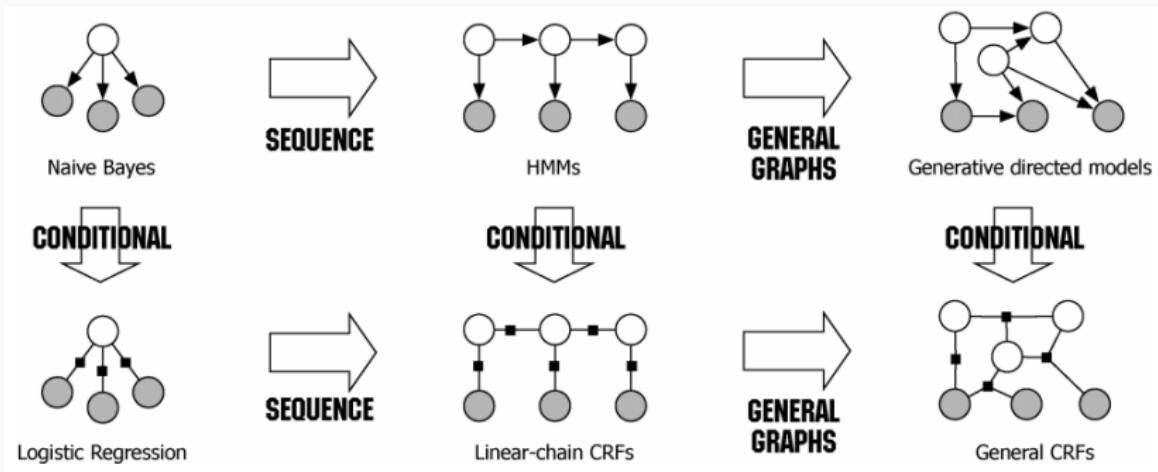
Порождающие и дискриминативные модели

- В результате мы будем хуже моделировать $p(x)$ (на которое в классификации нам наплевать), и лучше подходить к $p(y | x)$ (но и риск оверфиттинга выше).
- Другой пример порождающей модели – скрытая марковская модель: моделируем наблюдаемые $X = \{x_t\}_{t=1}^T$, предполагая, что есть скрытые состояния $Y = \{y_t\}_{t=1}^T$, и делаем предположения о независимости:

$$p(y, x) = \prod_{t=1}^T p(y_t | y_{t-1})p(x_t | y_t).$$

Порождающие и дискриминативные модели

- Так вот, CRF – это дискриминативные модели, и линейная CRF – это дискриминативная модель, соответствующая HMM:



Линейные CRF

- Чтобы прийти к linear chain CRF, сначала перепишем HMM в другом виде:

$$p(y, x) = \frac{1}{Z} \prod_{t=1}^T \exp \left(\sum_{i,j \in S} \theta_{ij} [y_t = i] [y_{t-1} = j] + \sum_{i \in S} \sum_{o \in O} \mu_{oi} [y_t = i] [x_t = o] \right),$$

где S – множество скрытых состояний, O – множество наблюдаемых, а в терминах HMM

$$\theta_{ij} = \log p(y_t = i \mid y_{t-1} = j), \\ \mu_{oi} = \log p(x = o \mid y = i), \quad Z = 1.$$

- Более того, верно и обратное: если распределение раскладывается как выше написано, то это HMM.

- И здесь тоже, как в логистической регрессии, можно ввести функции признаков (feature functions) $f_k(y_t, y_{t-1}, x_t)$:

$f_{ij}(y, y', x) = \llbracket y = i \rrbracket \llbracket y' = j \rrbracket$ для каждого перехода,

$f_{io}(y, y', x) = \llbracket y = i \rrbracket \llbracket x = o \rrbracket$ для каждой наблюдаемой.

- И теперь (k бегает и по (i, j) , и по (i, o))

$$p(y, x) = \frac{1}{Z} \prod_{t=1}^T \exp \left(\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right).$$

- Для выбранных f_k это в точности эквивалентно НММ.

Линейные CRF

- Можно теперь выразить условное распределение, и это уже будет linear chain CRF:

$$p(y | x) = \frac{\prod_{t=1}^T \exp \left(\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right)}{\sum_{y'} \prod_{t=1}^T \exp \left(\sum_{k=1}^K \theta_k f_k(y'_t, y'_{t-1}, x_t) \right)}.$$

- В общем виде linear chain CRF именно так и определяется:

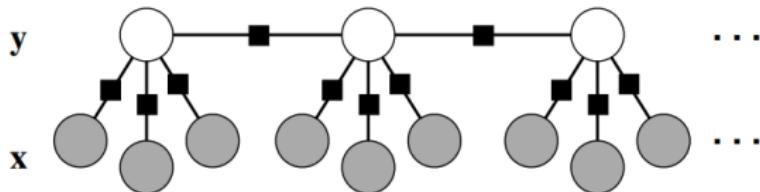
$$p(y | x) = \frac{1}{Z(x)} \prod_{t=1}^T \exp \left(\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right),$$

т.е. просто разрешим общий вид признаков и, может быть, несколько компонент у x_t .

Линейные CRF

- Linear chain CRF:

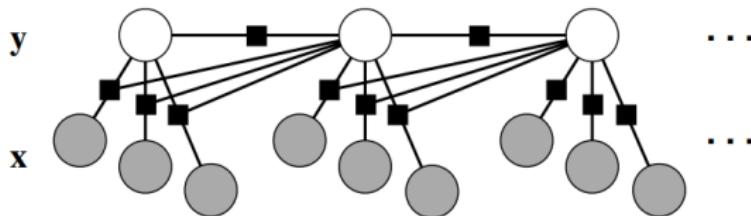
$$p(y | x) = \frac{1}{Z(x)} \prod_{t=1}^T \exp \left(\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right).$$



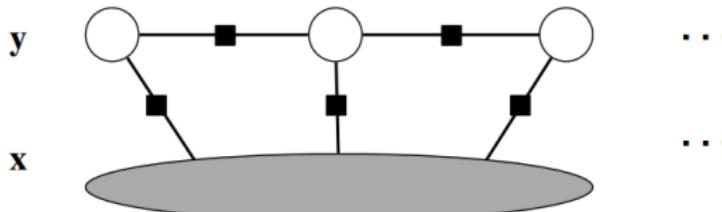
- Разложение $p(y | x) = \frac{1}{Z(x)} \prod_{t=1}^T \Psi_t(y_t, y_{t-1}, x_t)$, со специальным видом Ψ_t .
- Задача – оценить параметры θ из данных.
- Часто в NLP применяется: частеречная разметка, named entity recognition и т.п.

Линейные CRF

- Более общий случай, чем HMM; например, признак перехода может зависеть от наблюдаемой, просто добавим
 $f = \llbracket y_t = j \rrbracket \llbracket y_{t-1} = i \rrbracket \llbracket x_t = o \rrbracket$:



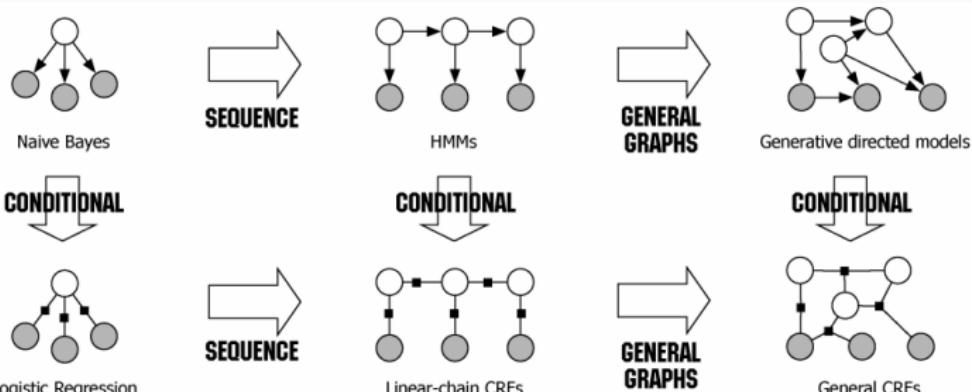
- А может зависеть от всех наблюдений сразу, $f_k(y_t, y_{t-1}, x)$:



Линейные CRF

- А совсем общая CRF – это модель, в которой условное распределение $p(y | x)$ раскладывается по какому-то графу, не обязательно цепи:

$$p(y | x) = \frac{1}{Z(x)} \prod_{a=1}^A \Psi_a(y_a, x_a).$$



- В чём отличие от просто общей формы ненаправленной модели?..

Линейные CRF

- О форме факторов:

- обычно мы обучаем CRF с линейными весами:

$$\log \Psi_a(y_a, x_a) = \exp \left(\sum_{k=1}^{K(A)} \theta_{ak} f_{ak}(y_a, x_a) \right),$$
 у каждого фактора свои веса;

- если x и y дискретные, то это вообще не ограничивает общности (возьмём f_{ak} по всем комбинациям);
 - часто параметры так или иначе связаны; например, в линейной цепи мы считаем, что $\Psi_t(y_t, y_{t-1}, x_t)$ одинаковые по всем t ;
 - можно определить clique templates для этого:

$$p(y | x) = \frac{1}{Z(x)} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(y_c, x_c; \theta_p), \text{ где}$$

$$\Psi_c(y_c, x_c; \theta_p) = \exp \left(\sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(x_c, y_c) \right);$$

- есть много расширений и вариантов с разными формами parameter tying.

- О признаках:
 - для дискретных наблюдаемых часто имеют смысл label-observation features:

$$f_{pk}(y_c, x_c) = \llbracket y_c = y'_c \rrbracket q_{pk}(x_c);$$

этот признак не ноль только для конкретной конфигурации y'_c , но для неё может быть любым; observation functions $q_{pk}(x_c)$ – например, «слово x_t начинается с большой буквы»;

- unsupported features: бывает, что признаки вообще не встречаются в данных, особенно когда их очень много (миллионы); например, « x_t = “если” и y_t = “CITYNAME”»; но они всё равно могут быть полезны, если дают отрицательный вес неподходящему ответу;

Линейные CRF

- О признаках:
 - edge-observation features: когда фактор перехода зависит от всех функций наблюдения: « x_t = “Новый” и y_t = “CITYNAME” и y_{t-1} = “CITYNAME”»;
 - если это слишком много, можно заменить на node-observation features: отдельно « x_t = “Новый” и y_t = “CITYNAME”», отдельно « y_t = “CITYNAME” и y_{t-1} = “CITYNAME”»;

Edge-observation features:

$$\begin{aligned} f(y_t, y_{t-1}, \mathbf{x}_t) &= q_m(\mathbf{x}_t) \mathbf{1}_{\{y_t=y\}} \mathbf{1}_{\{y_{t-1}=y'\}} & \forall y, y' \in \mathcal{Y}, \forall m \\ f(y_t, \mathbf{x}_t) &= q_m(\mathbf{x}_t) \mathbf{1}_{\{y_t=y\}} & \forall y \in \mathcal{Y}, \forall m \end{aligned}$$

Node-observation features:

$$\begin{aligned} f(y_t, y_{t-1}, \mathbf{x}_t) &= \mathbf{1}_{\{y_t=y\}} \mathbf{1}_{\{y_{t-1}=y'\}} & \forall y, y' \in \mathcal{Y} \\ f(y_t, \mathbf{x}_t) &= q_m(\mathbf{x}_t) \mathbf{1}_{\{y_t=y\}} & \forall y \in \mathcal{Y}, \forall m \end{aligned}$$

- О признаках:
 - отдельные метки START/STOP для первых/последних элементов последовательности.
 - важно не забывать, что в $f(y_t, y_{t-1}, x_t)$ среди x_t могут быть разные x : « x_{t+2} = “Уэльс” и y_t = “STATENAME”» (если есть такая фича, это, видимо, не Англия);
 - можно использовать менее общие признаки как регуляризатор (backoff): например, полезно включать и $\Psi_t(y_t, y_{t-1}, x_t)$, и $\Psi_t(y_t, x_t)$, хотя формально уже вторые не нужны;
 - можно добавлять как признаки результаты других моделей (более простых, или обученных на других датасетах и т.п.);
 - input-dependent structure: можно разрешать структуре графа зависеть от x ; например, можно поощрять, чтобы одно и то же слово имело одну и ту же метку.

Примеры

- Пример – named entity recognition (NER):

In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

- Например, датасет CoNLL2003 – люди (PER), организации (ORG), места (LOC), другие (MISC), и никаких (O). Девять видов меток:

$$\mathcal{Y} = \{\text{B-PER}, \text{I-PER}, \text{B-Loc}, \text{I-Loc}, \text{B-ORG}, \text{I-ORG}, \text{B-MISC}, \text{I-MISC}, \text{O}\}$$

Примеры

- Можно построить модель так:

t	y_t	\mathbf{x}_t
0	B-ORG	U.N.
1	O	official
2	B-PER	Ekeus
3	O	heads
4	O	for
5	B-LOC	Baghdad

- И ввести признаки label-label и label-word:

$$f_{ij}^{LL}(y_t, y_{t-1}, \mathbf{x}_t) = [\![y_t = i]\!][\![y_{t-1} = j]\!],$$

$$f_{iv}^{LW}(y_t, y_{t-1}, \mathbf{x}_t) = [\![y_t = i]\!][\![\mathbf{x}_t = v]\!].$$

- Всего 81 label-label и $9 \times 21249 = 191241$ label-word (в CoNLL2003), но большая часть бесполезны, конечно.

Примеры

- Но в реальности надо добавить выразительности:

t	y_t	\mathbf{x}_t
0	B-ORG	(⟨START⟩, U.N., official)
1	O	(U.N., official, Ekeus)
2	B-PER	(official, Ekeus, heads)
3	O	(Ekeus, heads, for)
4	O	(heads, for, Baghdad)
5	B-LOC	(for, Baghdad, ⟨END⟩)

- Три вида label-word теперь, и ещё label-observation можно добавить:

$$f_{iv}^{LW0}(y_t, y_{t-1}, \mathbf{x}_t) = \llbracket y_t = i \rrbracket \llbracket \mathbf{x}_{t0} = v \rrbracket,$$

$$f_{iv}^{LW1}(y_t, y_{t-1}, \mathbf{x}_t) = \llbracket y_t = i \rrbracket \llbracket \mathbf{x}_{t1} = v \rrbracket,$$

$$f_{iv}^{LW2}(y_t, y_{t-1}, \mathbf{x}_t) = \llbracket y_t = i \rrbracket \llbracket \mathbf{x}_{t2} = v \rrbracket,$$

$$f_{ib}^{LO}(y_t, y_{t-1}, \mathbf{x}_t) = \llbracket y_t = i \rrbracket q_b(\mathbf{x}_t).$$

Примеры

- Например, в (McCallum, Li, 2003):

W= v	$w_t = v$	$\forall v \in \mathcal{V}$
T= j	part-of-speech tag for w_t is j (as determined by an automatic tagger)	$\forall \text{POS tags } j$
P=I- j	w_t is part of a phrase with syntactic type j (as determined by an automatic chunker)	
Capitalized	w_t matches [A-Z] [a-z] +	
Allcaps	w_t matches [A-Z] [A-Z] +	
EndsInDot	w_t matches [^\.] +.*\.	
	w_t contains a dash	
	w_t matches [A-Z] + [a-z] + [A-Z] + [a-z]	
Acro	w_t matches [A-Z] [A-Z\\.] *\\.[A-Z\\.] *	
Stopword	w_t appears in a hand-built list of stop words	
CountryCapital	w_t appears in list of capitals of countries	
:	many other lexicons and regular expressions	
$q_k(\mathbf{x}, t + \delta)$ for all k and $\delta \in [-1, 1]$		

Примеры

- Например, в (McCallum, Li, 2003):

t	y_t	Active observation functions
1	B-ORG	P=I-NP@1 W=(START)@-1 INITCAP P=I-NP T=NNP T=NN@1 ACRO ENDSINDOT W=official@1 W=U.N.
2	O	P=I-NP@1 INITCAP@1 P=I-NP T=JJ@1 CAPITALIZED@1 T=NNP@-1 P=I-NP@-1 INITCAP@-1 T=NN ENDSINDOT@-1 ACRO@-1 W=official W=U.N.@-1 W=Ekeus@1
3	B-PER	P=I-NP@1 INITCAP P=I-NP P=I-NP@-1 CAPITALIZED T=JJ T=NN@-1 T=NNS@1 W=official@-1 W=heads@1 W=Ekeus
4	O	P=I-NP P=I-NP@-1 INITCAP@-1 STOPWORD@1 T=JJ@-1 CAPITALIZED@-1 T=IN@1 P=I-PP@1 T=NNS W=for@1 W=heads W=Ekeus@-1
5	O	T=NNP@1 P=I-NP@1 INITCAP@1 LOC@1 CAPITALIZED@1 P=I-NP@-1 STOPWORD COUNTRYCAPITAL@1 P=I-PP T=IN T=NNS@-1 W=for W=Baghdad@1 W=heads@-1
6	B-LOC	INITCAP P=I-NP T=NNP CAPITALIZED STOPWORD@-1 T=.=@1 P=O@1 PUNC@1 W=(END)@1 COUNTRYCAPITAL T=IN@-1 P=I-PP@-1 W=for@-1 W=Baghdad

Примеры

- Теперь к нашему основному примеру – сегментации, т.е. разметке картинки на семантические части.
- Формально $\mathbf{x} = (x_1, \dots, x_T)$, картинка размера $\sqrt{T} \times \sqrt{T}$. Предсказываем $\mathbf{y} = (y_1, \dots, y_T)$, сегментацию.
- Куча признаков в классическом CV:
 - гистограммы интенсивностей пикселей, например в окрестности 5×5 ;
 - градиенты картинки;
 - texton-признаки, SIFT-признаки и т.п. (может быть, позже).
- Как их всех в CRF добавить?

Примеры

- Базовая идея – по-прежнему MRF:

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{(i,j) \in \mathcal{N}} \Psi(y_i, y_j), \quad p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}) \prod_{i=1}^T p(x_i | y_i).$$

- Но трудно добавить признаки по подмножествам пикселей с предыдущего слайда, потому что $p(\mathbf{x} | \mathbf{y})$ получается сложная.
- А в CRF всё с этим проще: мы разрешаем факторам зависеть от произвольных признаков (observation functions) всей картинки.

Примеры

- Например, пусть $q(x_i)$ – вектор признаков участка вокруг x_i , $\nu(x_i, x_j)$ – признаки пары участков (их похожесть-непохожесть), например набор элементов матрицы $q(x_i)q(x_j)^\top$, и из них определяем CRF с такими observation functions:

$$f_m(y_i, x_i) = \mathbf{1}_{\{y_i=m\}} q(x_i) \quad \forall m \in \{0, 1\}$$

$$g_{m,m'}(y_i, y_j, x_i, x_j) = \mathbf{1}_{\{y_i=m\}} \mathbf{1}_{\{y_j=m'\}} \nu(x_i, x_j) \quad \forall m, m' \in \{0, 1\}$$

$$f(y_i, x_i) = \begin{pmatrix} f_0(y_i, x_i) \\ f_1(y_i, x_i) \end{pmatrix}$$

$$g(y_i, y_j, x_i, x_j) = \begin{pmatrix} g_{00}(y_i, y_j, x_i, x_j) \\ g_{01}(y_i, y_j, x_i, x_j) \\ g_{10}(y_i, y_j, x_i, x_j) \\ g_{11}(y_i, y_j, x_i, x_j) \end{pmatrix}$$

Примеры

- Конкретный пример функций:

$$\nu(x_i, x_j) = \exp(-\beta(x_i - x_j)^2),$$
$$g(y_i, y_j, x_i, x_j) = [\![y_i \neq y_j]\!] \nu(x_i, x_j).$$

- И получаем CRF

$$p(y | x) = \frac{1}{Z(x)} \exp \left(\sum_{i=1}^T \boldsymbol{\theta}^\top f(y_i, x_i) + \alpha \sum_{(i,j) \in \mathcal{N}} \boldsymbol{\lambda}^\top g(y_i, y_j, x_i, x_j) \right),$$

где $\alpha, \boldsymbol{\theta}, \boldsymbol{\lambda}$ – параметры.

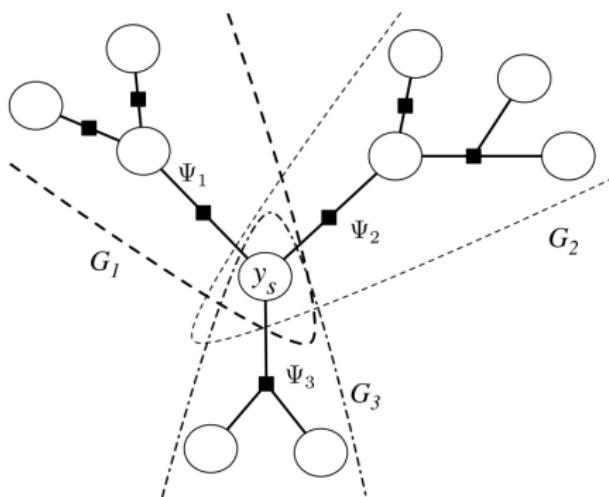
- Теперь встаёт вопрос: а как эту всю красоту обучить-то?

Вывод в CRF

- Вывод в CRF состоит из двух похожих задач:
 - применить CRF, т.е. найти метки $y^* = \arg \max_y p(y | x)$;
 - чтобы оценить параметры, надо найти маргиналы $p(y_t | x)$ и $p(y_t, y_{t-1} | x)$ (потом увидим почему).
- Всё это чертовски напоминает HMM для linear chain CRF и просто вывод в графических моделях для CRF общего вида.

Выход в CRF

- Linear chain CRF – те же forward-backward, Viterbi, $\alpha_t(j)$, $\beta_t(j)$, $\gamma_t(j)$...
- В общем виде – тот же belief propagation



Вывод в CRF

- А когда так не получается, те же МСМС-методы, вариационные приближения...
- Но с CRF надо иметь в виду, что:
 - есть обычно большая разреженность в признаках и значениях факторов, которую надо бы использовать;
 - второе замечание – как справиться с underflow при выводе? можно считать в логарифмах:

$$\log \alpha_t(j) = \bigoplus_{i \in S} (\log \Psi_t(j, i, x_t) + \log \alpha_{t-1}(i)),$$

но разве это проще? вроде бы при подсчёте
 $a \oplus b = \log(e^a + e^b)$ всё равно тот же underflow? hint: нет.

Вывод в CRF

- А как оценивать параметры CRF $\theta = \{\theta_k\}_{k=1}^K$?
- Максимальное правдоподобие: по данным $D = \{x^{(i)}, y^{(i)}\}_{i=1}^N$ надо максимизировать условное правдоподобие

$$\ell(\theta) = \sum_{i=1}^N \log p(y^{(i)} | x^{(i)}; \theta).$$

- Для linear chain CRF, например,

$$\ell(\theta) = \sum_{i=1}^N \sum_{t=1}^T \sum_{k=1}^K \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \sum_{i=1}^N \log Z(x^{(i)}).$$

- Можно ещё добавить регуляризацию, L_2 или L_1 на θ .
- Как максимизировать?

Вывод в CRF

- Возьмём производную:

$$\begin{aligned}\frac{\partial \ell}{\partial \theta_k} = & \sum_{i=1}^N \sum_{t=1}^T f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \\ & - \sum_{i=1}^N \sum_{t=1}^T \sum_{y,y'} f_k(y, y', \mathbf{x}_t^{(i)}) p(y, y' | \mathbf{x}^{(i)}) - \frac{\partial \Omega}{\partial \theta_k}.\end{aligned}$$

- Первое слагаемое – ожидание f_k в эмпирическом распределении $p_d(\mathbf{y}, \mathbf{x}) = \frac{1}{N} \sum_{i=1}^N [\mathbf{y} = \mathbf{y}^{(i)}] [\mathbf{x} = \mathbf{x}^{(i)}]$.
- Второе слагаемое – ожидание f_k в распределении модели $p(\mathbf{y} | \mathbf{x}; \boldsymbol{\theta}) p_d(\mathbf{x})$.
- Если найдём максимум, они совпадут – приятное свойство.
- Ещё приятно, что $\ell(\boldsymbol{\theta})$ вогнута, т.к. $g(\mathbf{x}) = \log \sum_i \exp x_i$ – выпуклые функции; если ещё добавить строго выпуклый регуляризатор вроде L_2 , у ℓ будет единственный глобальный оптимум.

Вывод в CRF

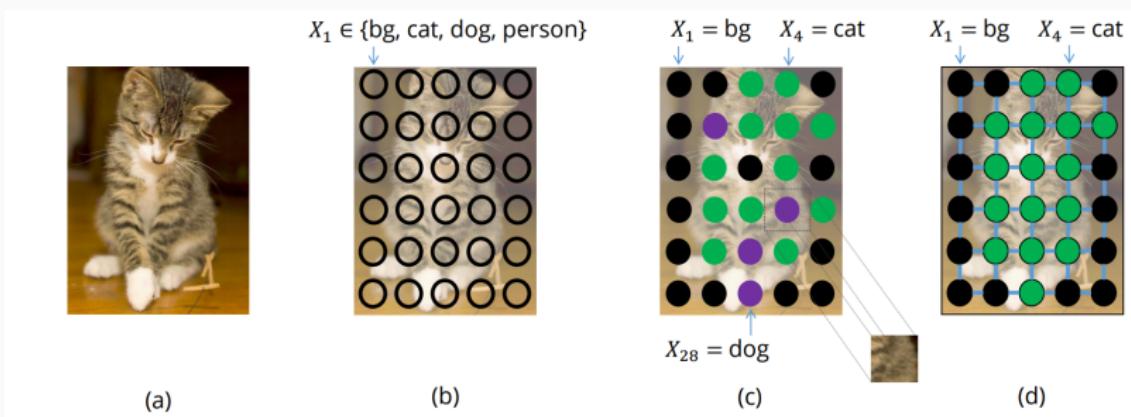
- Можно применять стандартные алгоритмы: метод Ньютона, его ускоренную версию L-BFGS...
- И для CRF общего вида то же самое на самом деле:

$$\ell(\boldsymbol{\theta}) = \sum_{C_p \in \mathcal{C}} \sum_{\Psi_c \in C_p} \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(x_c, y_c) - \log Z(x).$$

- Тоже всё вогнутое, можно использовать методы второго порядка.
- Но проблема: для каждого шага надо считать $p(y, y' | x^{(i)})$, т.е. делать вывод в CRF для каждого тренировочного примера.
- Если так не получается, то надо делать стохастический градиентный подъём.

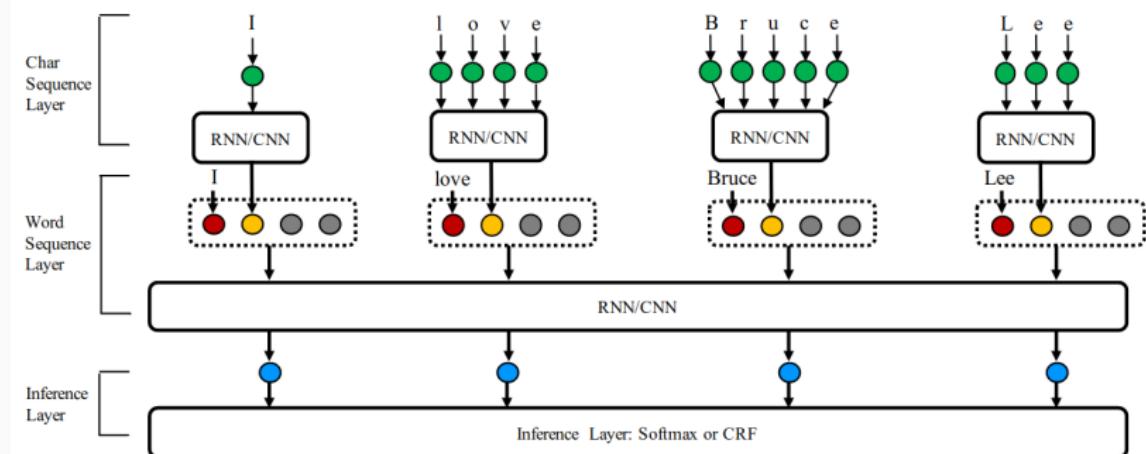
Deep learning + CRF

- И теперь можно пытаться вставлять в CRF признаки из глубоких сетей.
 - Например, предскажем метку каждого пикселя классификатором (на основе всей картинки или близлежащей части), а потом вставим в CRF:



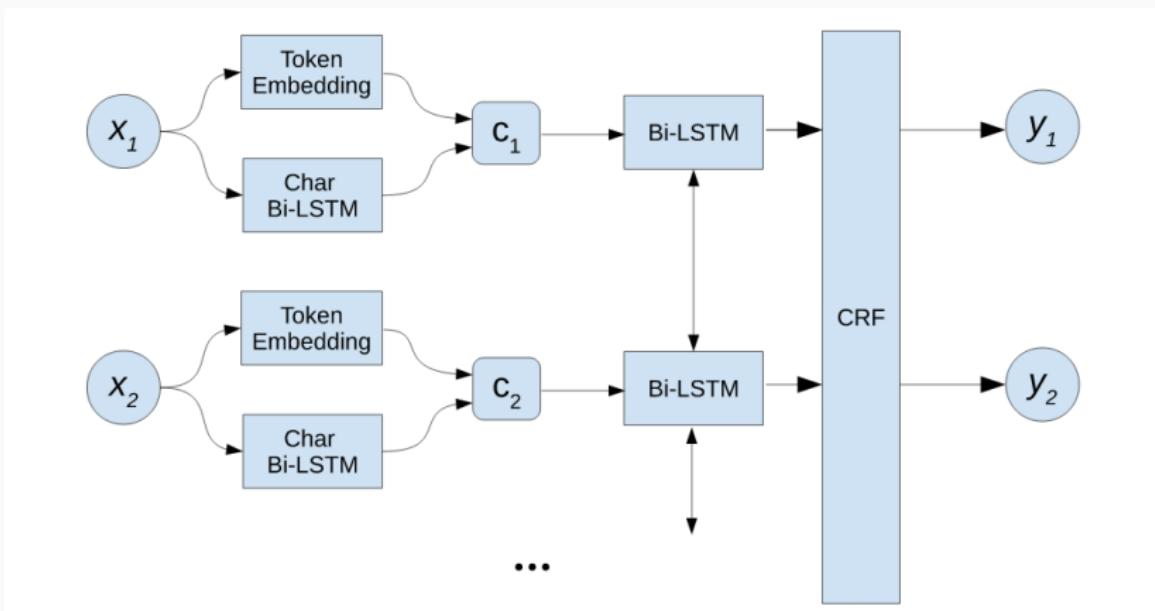
Deep learning + CRF

- Или вот NER – там RNN+CRF архитектуры до сих пор актуальны, в том числе в архитектурах вида «учитель-ученик», чтобы трансформеры не гонять:



Deep learning + CRF

- Вот, например, как NER от DeepPavlov работал (Anh et al., 2017):

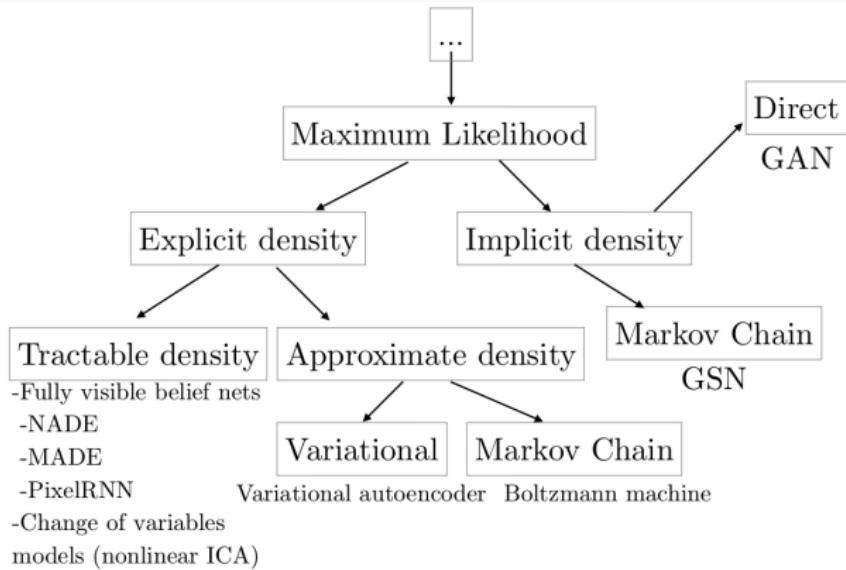


- Structured prediction: более общий контекст, когда мы пытаемся сделать структурированные предсказания.
- CRF – один вид, бывают ещё structured SVM, например, которые вообще не очень вероятностные.
- Есть приложения в компьютерном зрении.

Порождающие модели в глубоком обучении

Порождающие модели в DL

- Теперь возвращаемся к глубокому обучению
- Общая таксономия (Goodfellow, 2014):



Порождающие модели в DL

- Если можем явно выразить порождающую плотность (explicit), то, например, FVBN (fully visible belief networks):

$$p_{\text{model}}(\mathbf{x}) = \prod_{i=1}^n p_{\text{model}}(x_i \mid x_1 \dots, x_{i-1}).$$

- А с одномерными распределениями уж как-нибудь разберёмся, например промоделируем нейронной сетью.
- Появились в конце 90-х (Frey et al., 1996; Frey, 1998).

Порождающие модели в DL

- (Germain et al., 2015): MADE, Masked Autoencoder for Distribution Estimation
- Пример авторегрессионной модели, основанной на автокодировщиках:

$$\ell(\mathbf{x}) = \sum_{d=1}^D (-x_d \log \hat{x}_d - (1 - x_d) \log(1 - \hat{x}_d)) ,$$

где $\hat{\mathbf{x}}$ – бинарный выход, порождённый моделью.

- Теперь вопрос: можно ли сделать так, чтобы выход интерпретировался как вероятность?

Порождающие модели в DL

- Можно! Начинаем с той же идеи:

$$p_{\text{model}}(\mathbf{x}) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1}) = p(x_i | \mathbf{x}_{<i}).$$

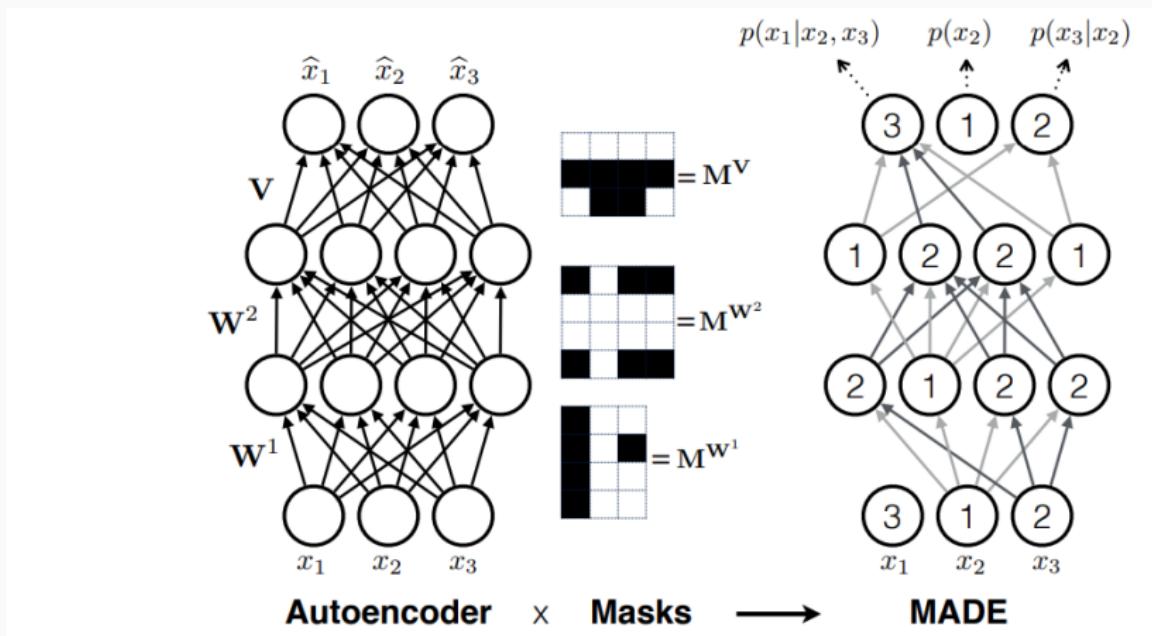
- И теперь возьмём $\hat{x}_d = p(x_i = 1 | \mathbf{x}_{<i})$, и получится правдоподобие

$$-\log p(\mathbf{x}) = \sum_{d=1}^D (-x_d \log p(x_d = 1 | \mathbf{x}_{<d}) - (1 - x_d) \log p(x_d = 0 | \mathbf{x}_{<d})).$$

- Осталось позаботиться о том, чтобы на следующие пиксели не слишком смотреть.

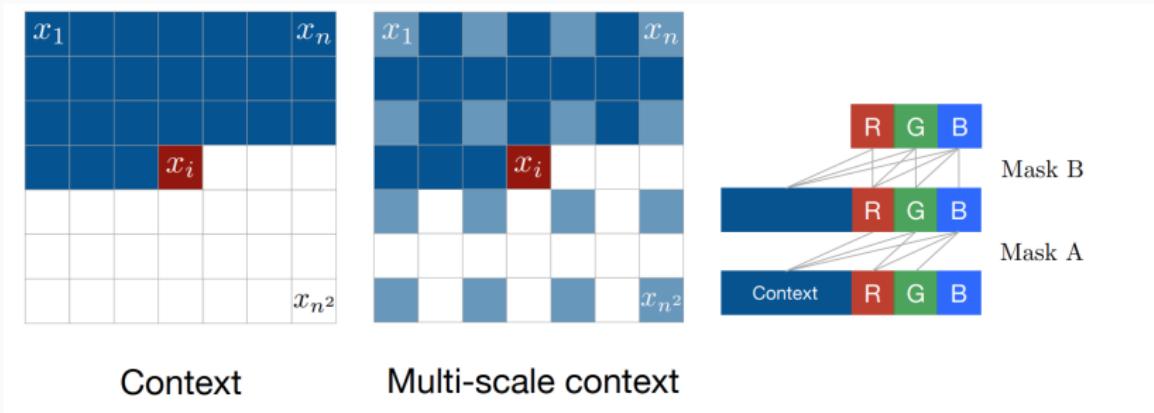
Порождающие модели в DL

- А этого можно достичь масками, причём их можно выбирать так, чтобы в случайном порядке раскладывать выход:



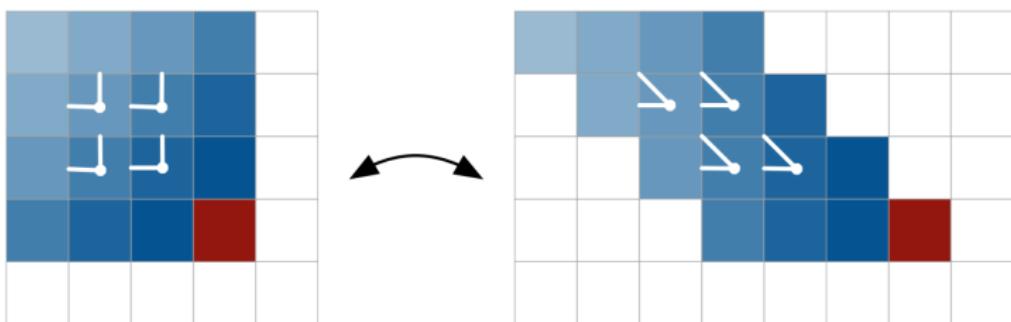
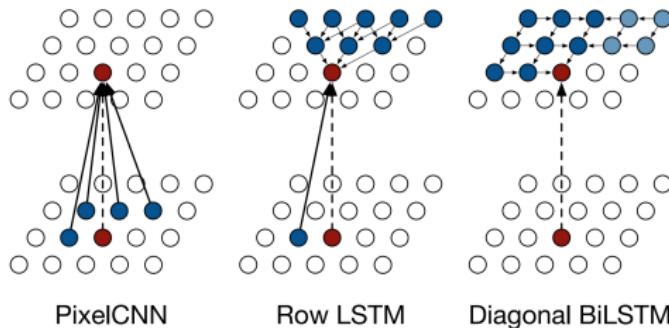
Порождающие модели в DL

- (van den Oord, 2016a; 2016b): давайте порождать картинку пиксель за пикселем! Моделируем сетью распределение $p(x_i | x_1, \dots, x_{i-1})$



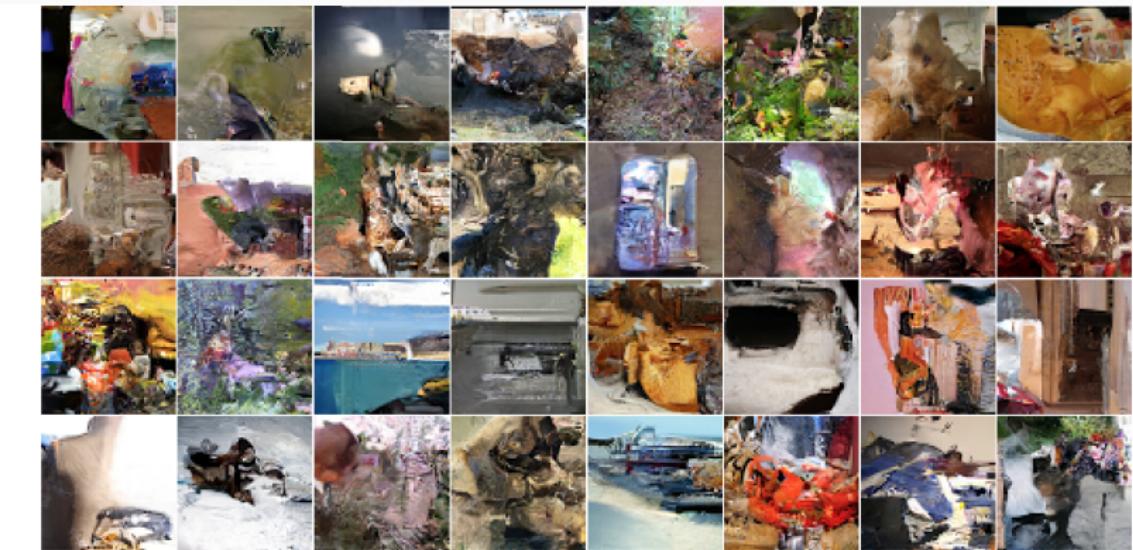
Порождающие модели в DL

- Можно моделировать свёрточной сетью или рекуррентной:



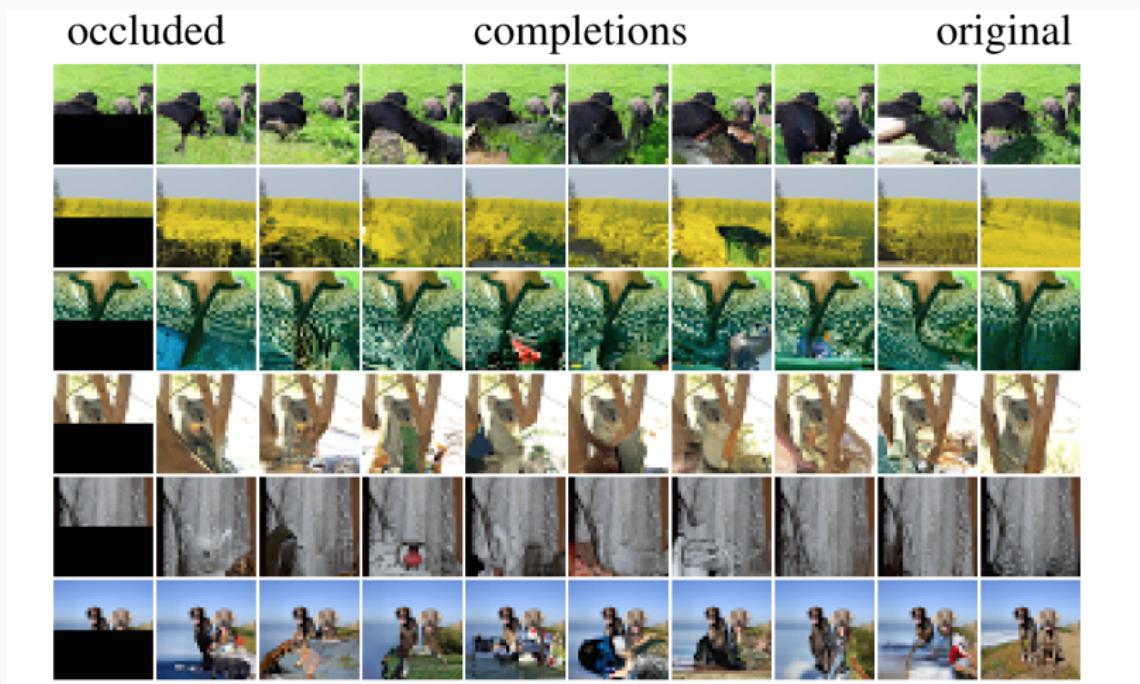
Порождающие модели в DL

- (van den Oord, 2016a) – PixelRNN:



Порождающие модели в DL

- (van den Oord, 2016a) – PixelRNN:



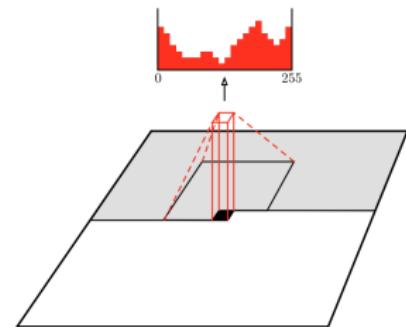
Порождающие модели в DL

- (van den Oord, 2016b) – PixelCNN; точно так же

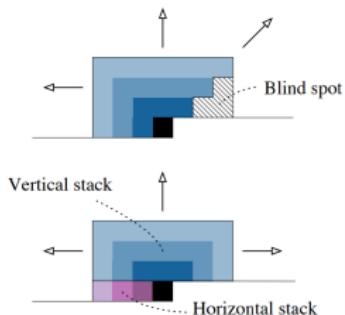
$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1}),$$

но теперь будем свёрточной сетью моделировать.

- Нужны маски, чтобы не забегать вперёд:

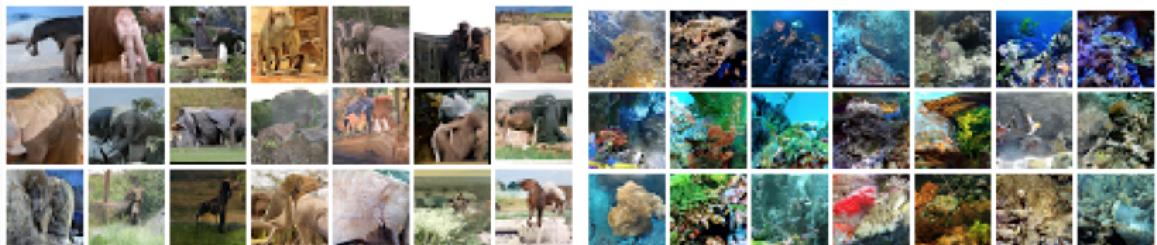


1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0



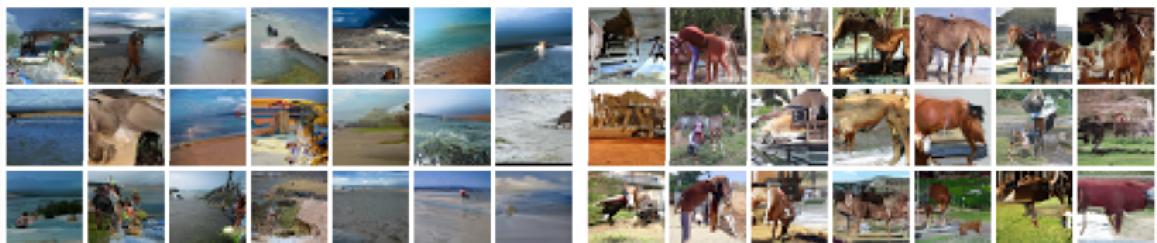
Порождающие модели в DL

- (van den Oord, 2016b) – PixelCNN:



African elephant

Coral Reef

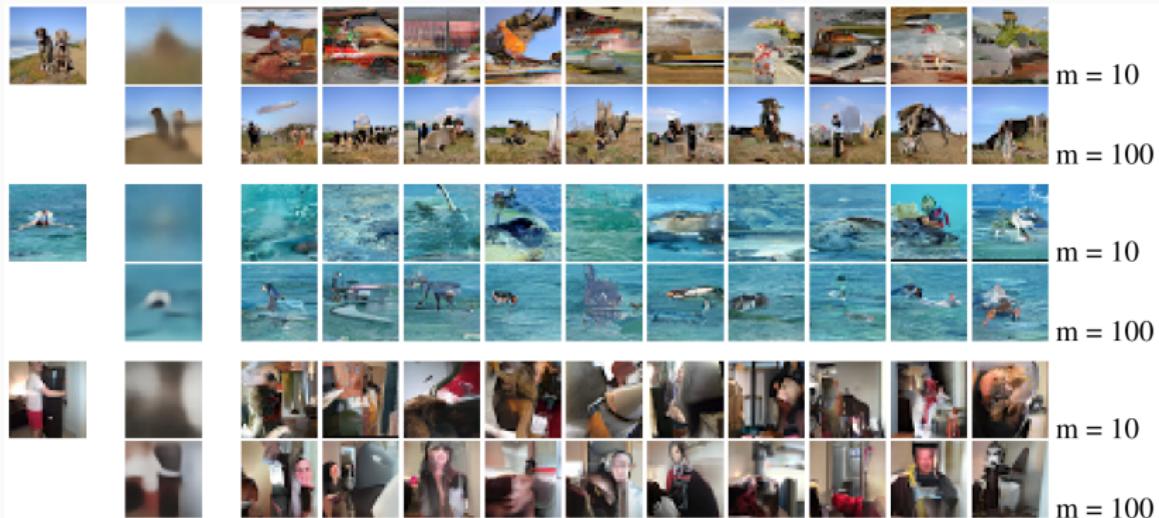


Sandbar

Sorrel horse

Порождающие модели в DL

- (van den Oord, 2016b) – PixelCNN autoencoder:



WaveNet и порождение звука

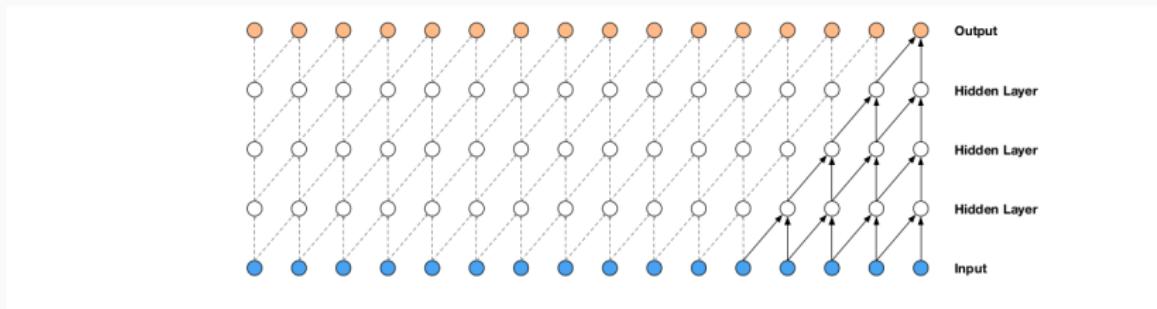
- Другой современный пример FVBN – WaveNet (van den Oord et al., 2016).
- Мы обычно мало говорим о звуке в DL, да и работ мало, WaveNet одна из немногих.
- Как породить, например, человеческую речь?
- Основная идея – будем моделировать условное распределение

$$p(\mathbf{x} \mid \mathbf{h}) = \prod_{t=1}^T p(x_t \mid x_1, \dots, x_{t-1}, \mathbf{h}).$$

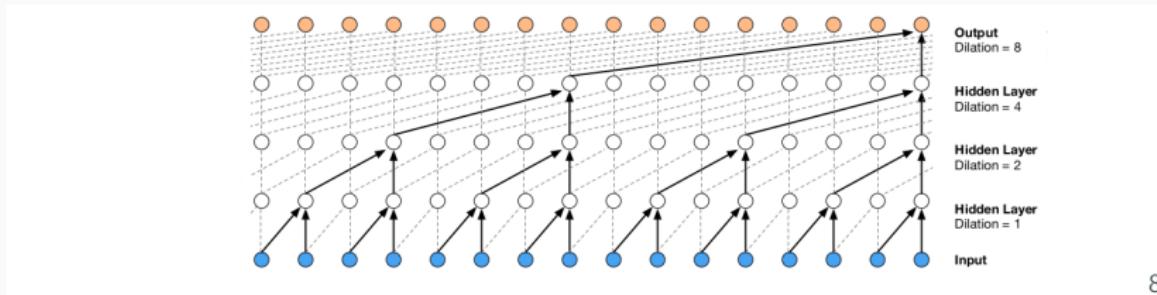
- На звуковых данных полезно делать одномерные свёртки, но свёртки не должны «забегать вперёд» по времени...

WaveNet и порождение звука

- ...поэтому в WaveNet каузальные свёртки (causal convolutions):

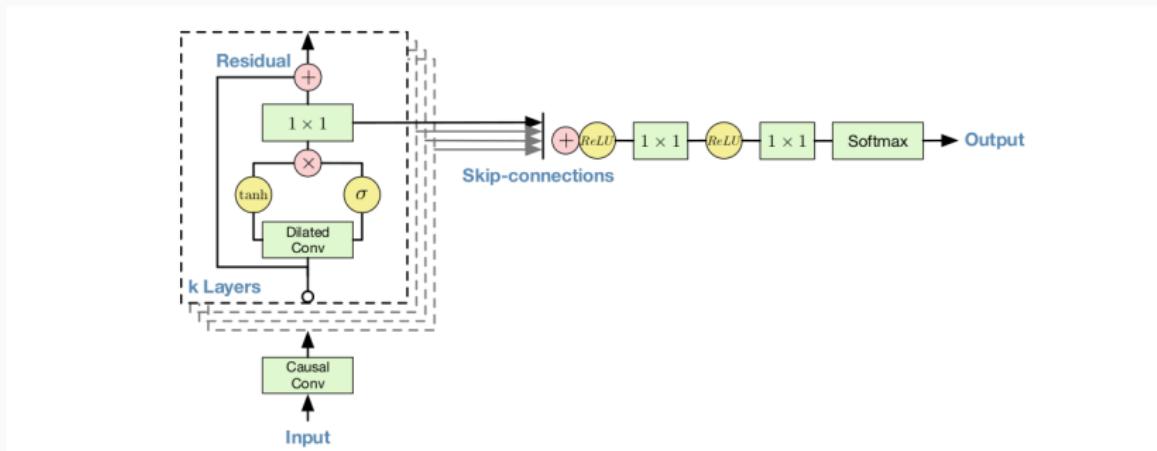


- Их лучше прореживать:



WaveNet и порождение звука

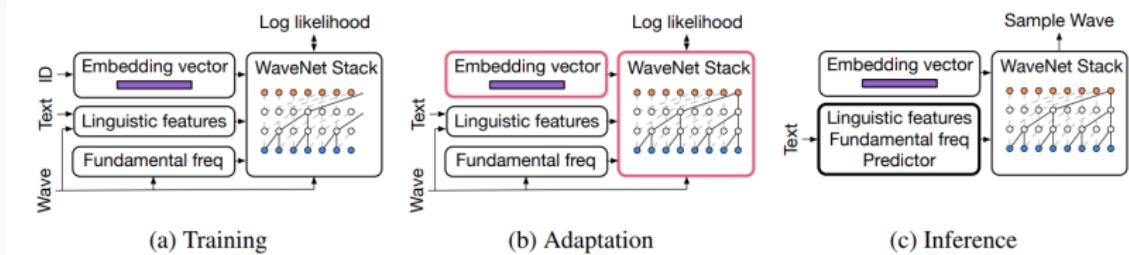
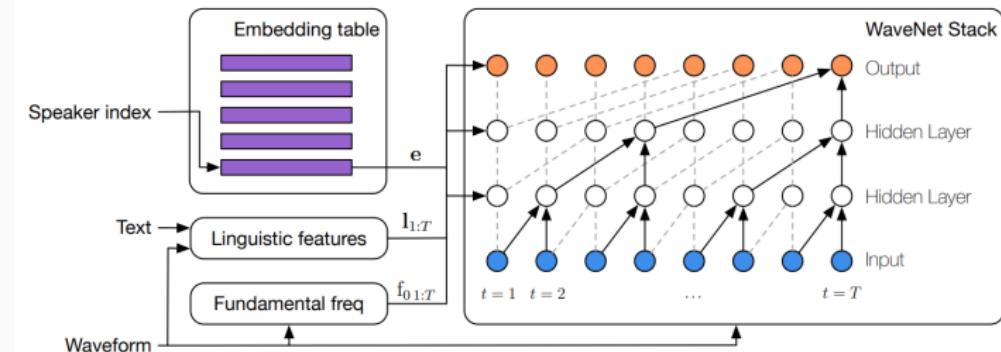
- А дальше архитектура с трюками, которые мы уже знаем – residual connections, skip-layer connections...



- Получалось очень хорошо ещё в 2016:
<https://deepmind.com/blog/wavenet-generative-model-raw-audio/>

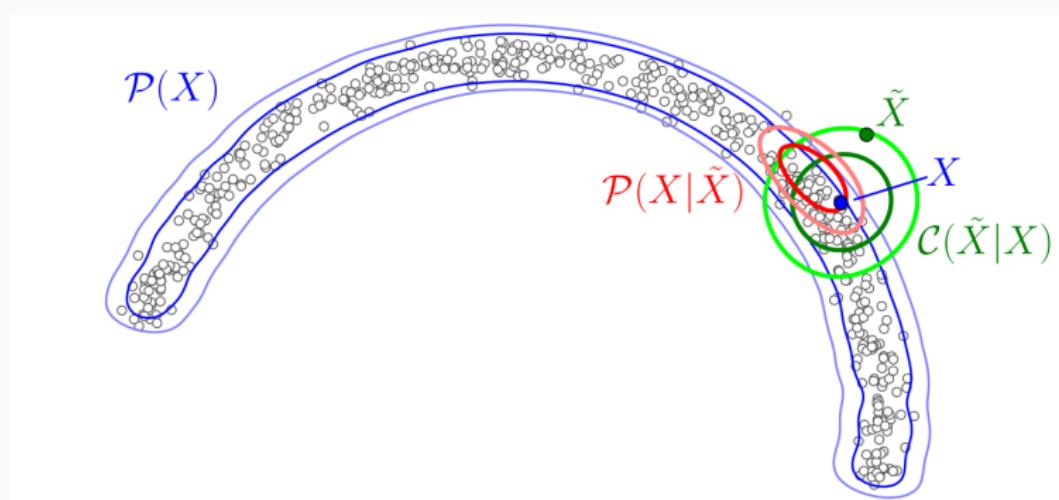
WaveNet и порождение звука

- Сейчас есть несколько важных работ про latent representations для речи, основанные на WaveNet
- (Chen et al., 2019): adaptive text-to-speech



GSN как пример неявных моделей

- А для неявных (implicit) порождающих моделей мы моделируем обычно процесс сэмплирования.
- Сэмплировать можно марковской цепью – если её моделировать нейронной сетью, получится Generative Stochastic Network (Alain et al., 2015):



- Но мы пойдём другим путём...

Спасибо!

Спасибо за внимание!

