ОНЛАЙН-ОБРАЗОВАНИЕ

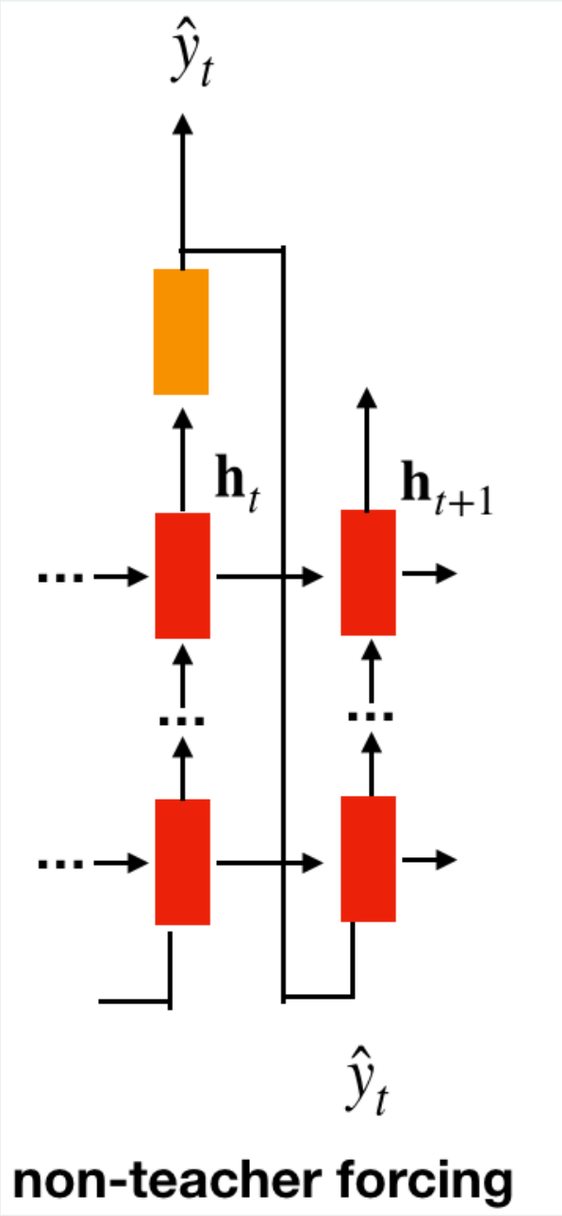
Рекуррентные сети. Обзор.

Михаил Степанов
Преподаватель

План на сегодня

1. **Учителю виднее!**
2. Профессор против Учителя
3. SeqGAN
4. Gumbel-softmax trick
5. Grammar VAE

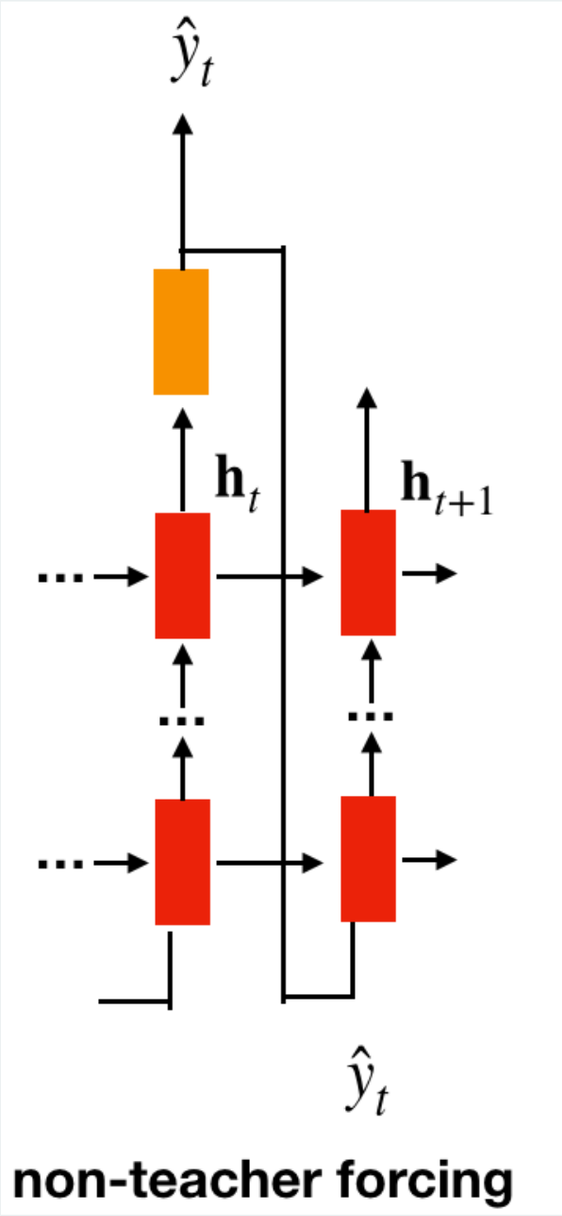
Teacher Forcing



non-teacher forcing

При обучении рекуррентной сети вход очередного шага является выходом предыдущего шага.

Teacher Forcing

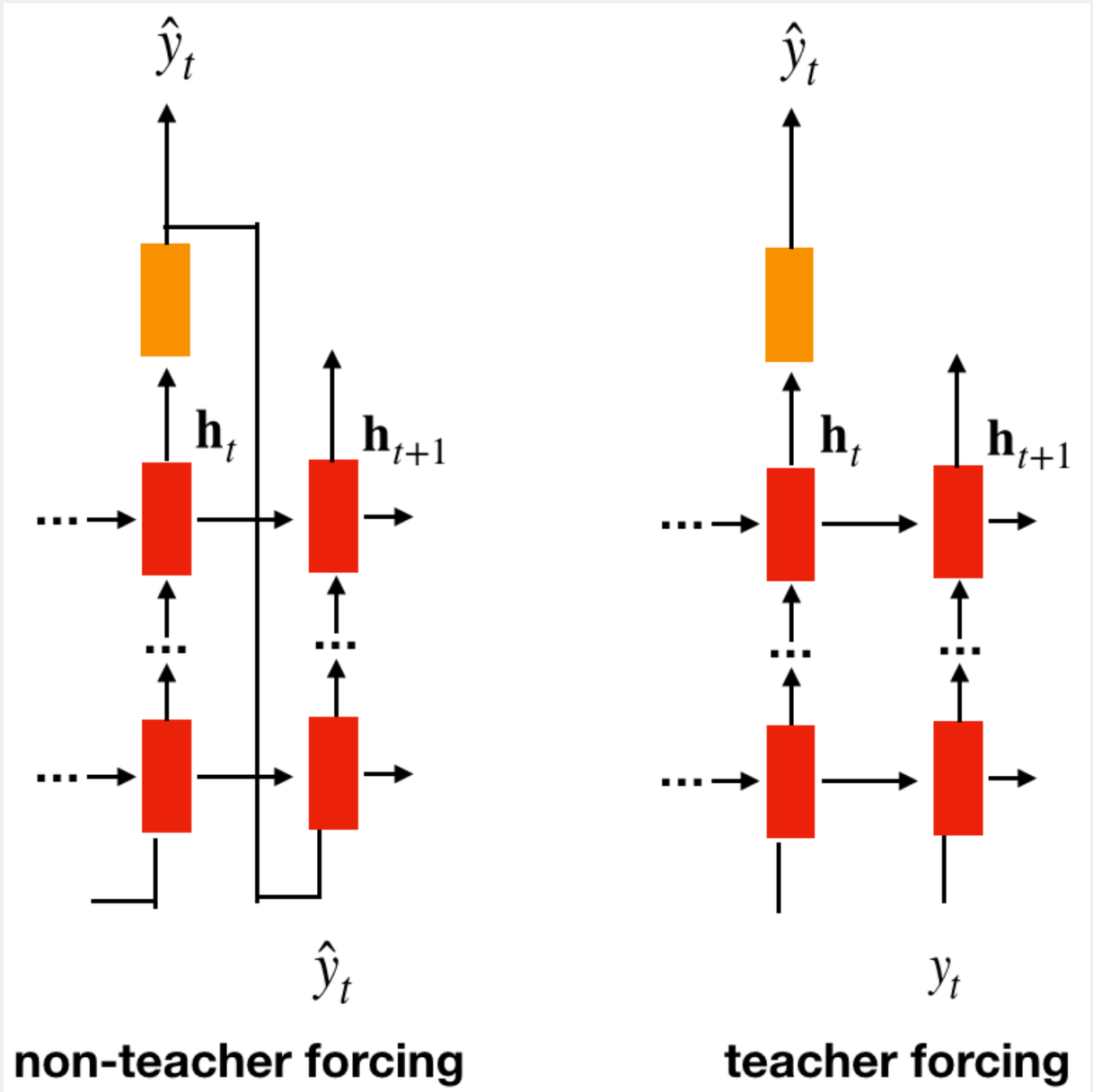


non-teacher forcing

При обучении рекуррентной сети вход очередного шага является выходом предыдущего шага.

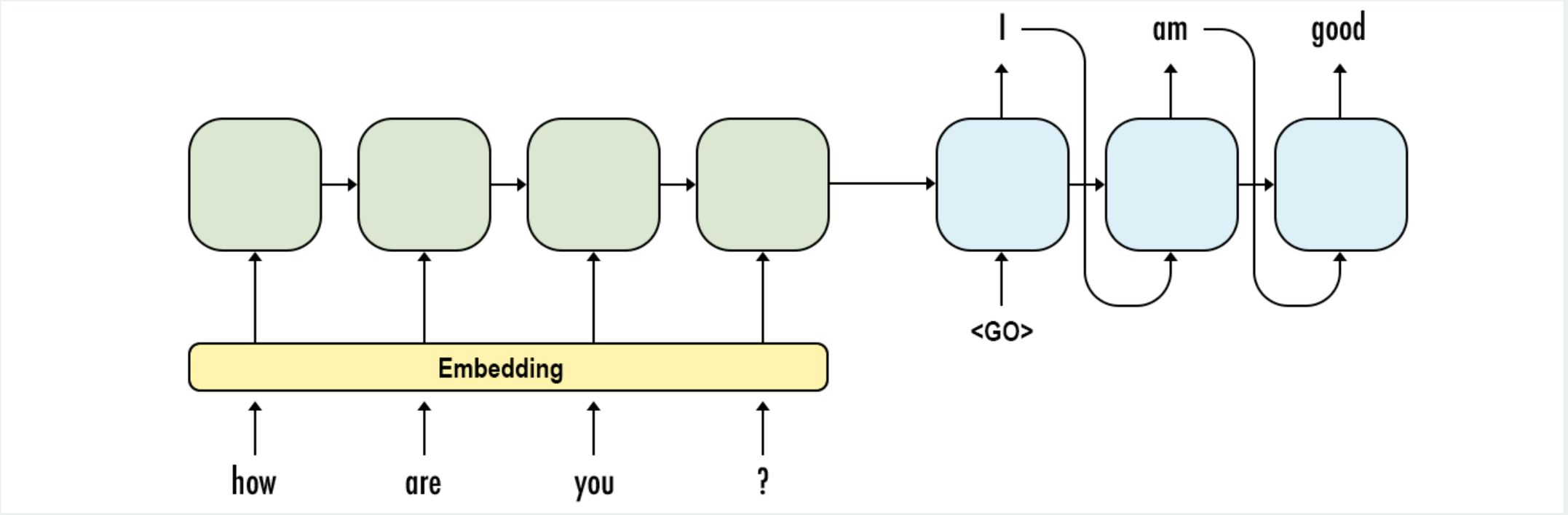
Однако если сеть ошиблась уже в первом символе, дальше ошибка будет только расти.

Teacher Forcing



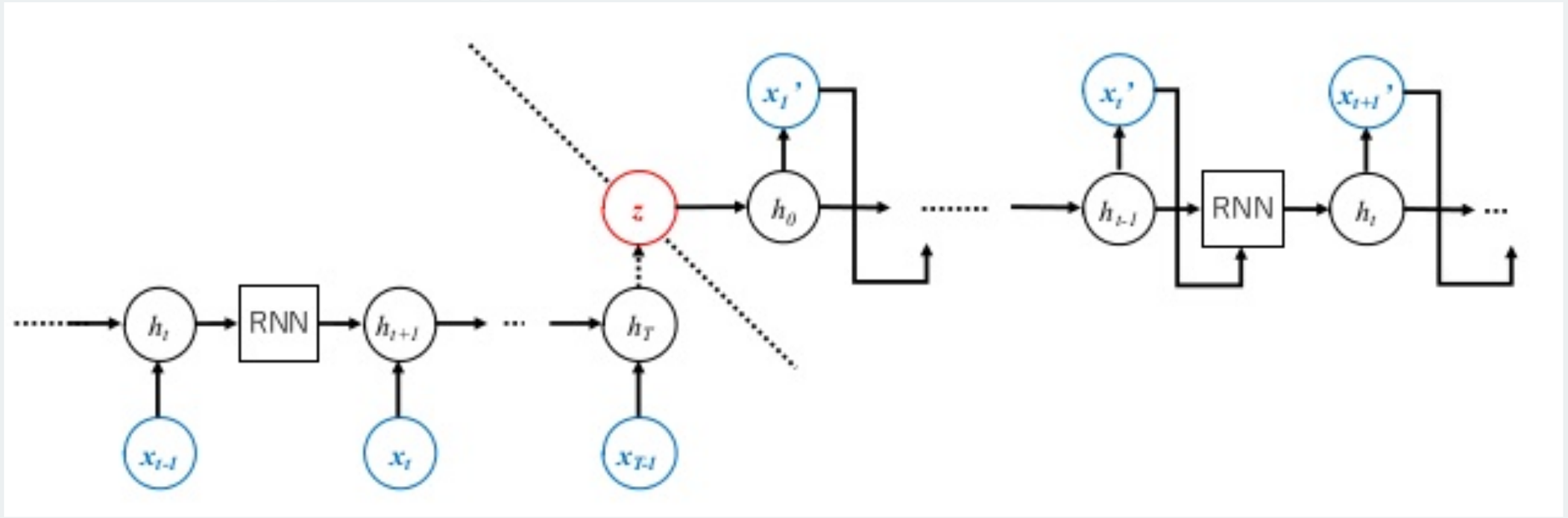
Поэтому при обучении генеративных рекуррентных моделей часто используют технику Teacher Forcing. Если мы знаем какой должна быть последовательность на выходе, то на очередном шаге будем подавать «правильный» символ вместо сгенерированного.

Инференс



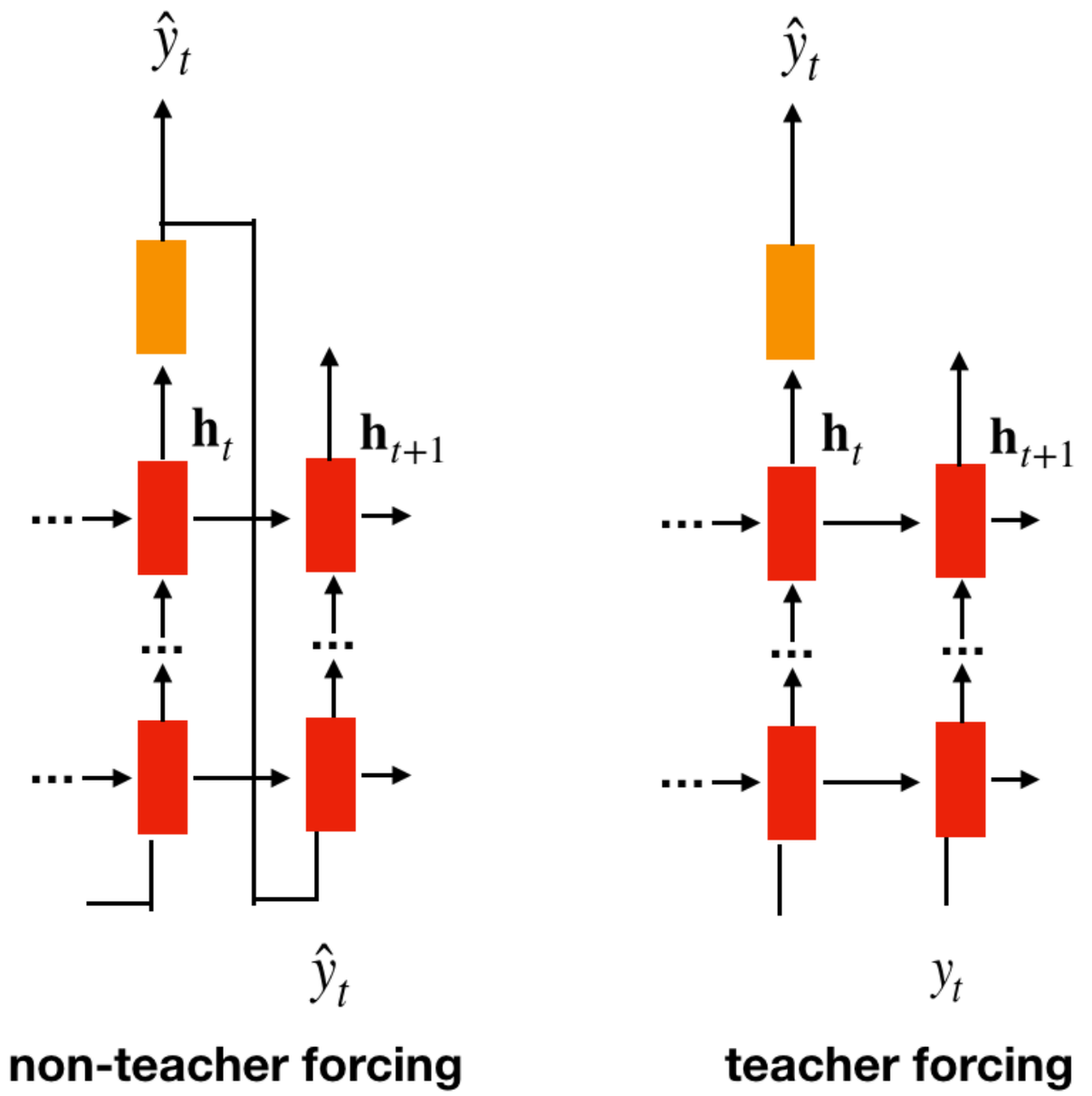
При обучении модели используется подход Teacher Forcing, однако во время использования мы генерируем ответную последовательность подавая на вход очередного шага то что на самом деле было сгенерировано сетью на предыдущем шаге (Free Running).

VRAE: пример модели



Как не трудно догадаться, VRAE — это Seq2Seq с репараметризацией и KL-дивергенцией на латентном слое. Или просто VAE с рекуррентными Энкодером и Декодером. Аналогичным образом можно получить состязательный автокодировщик и условные версии этих моделей.

Professor Forcing



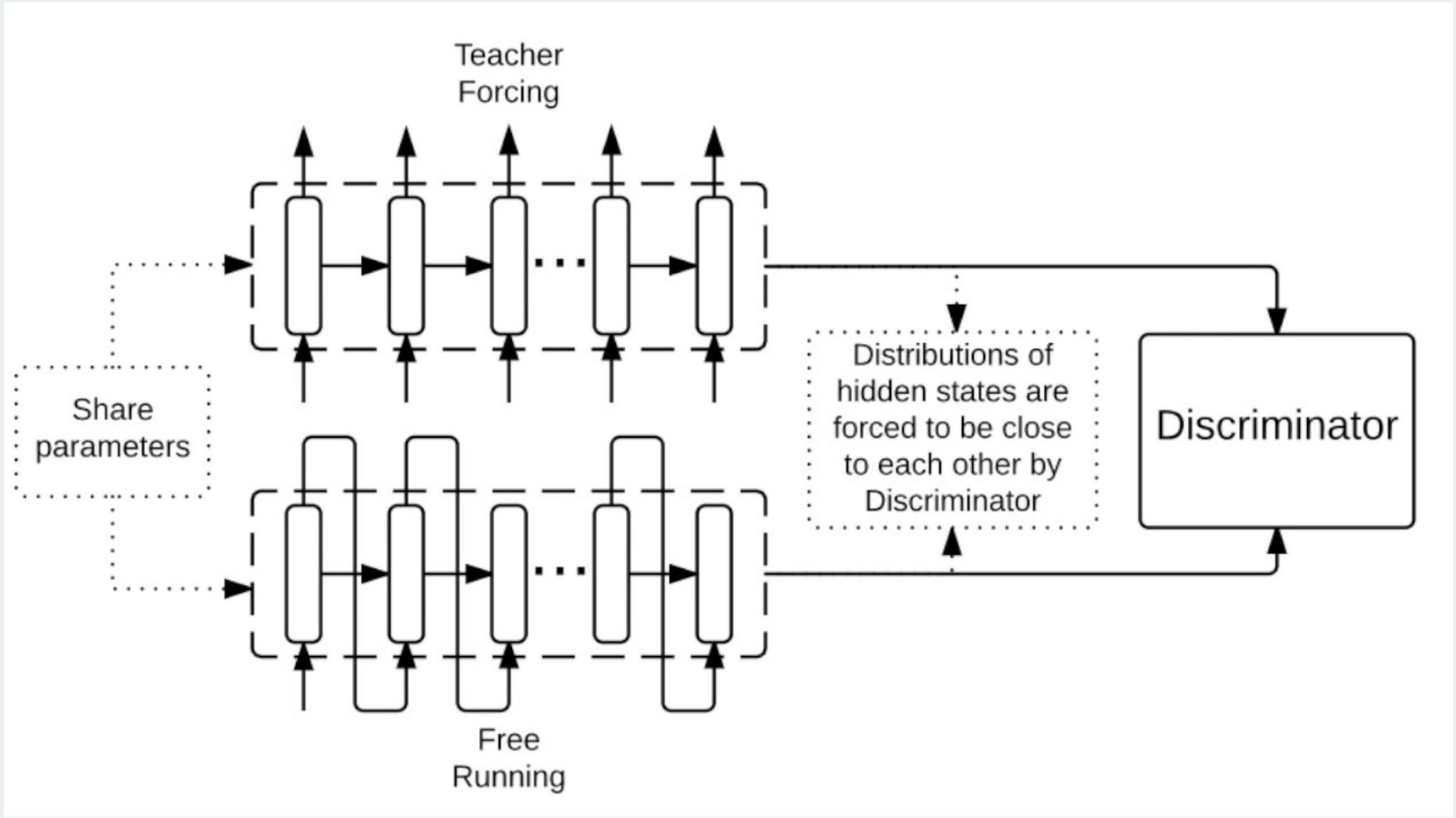
Одна из проблем Teacher Forcing заключается в том, что состояния ячеек во время free running могут сильно отличаться от состояний при использовании teacher forcing. Т.е. во время генерации новых примеров мы работаем в другой области пространства.

Как это исправить?

План на сегодня

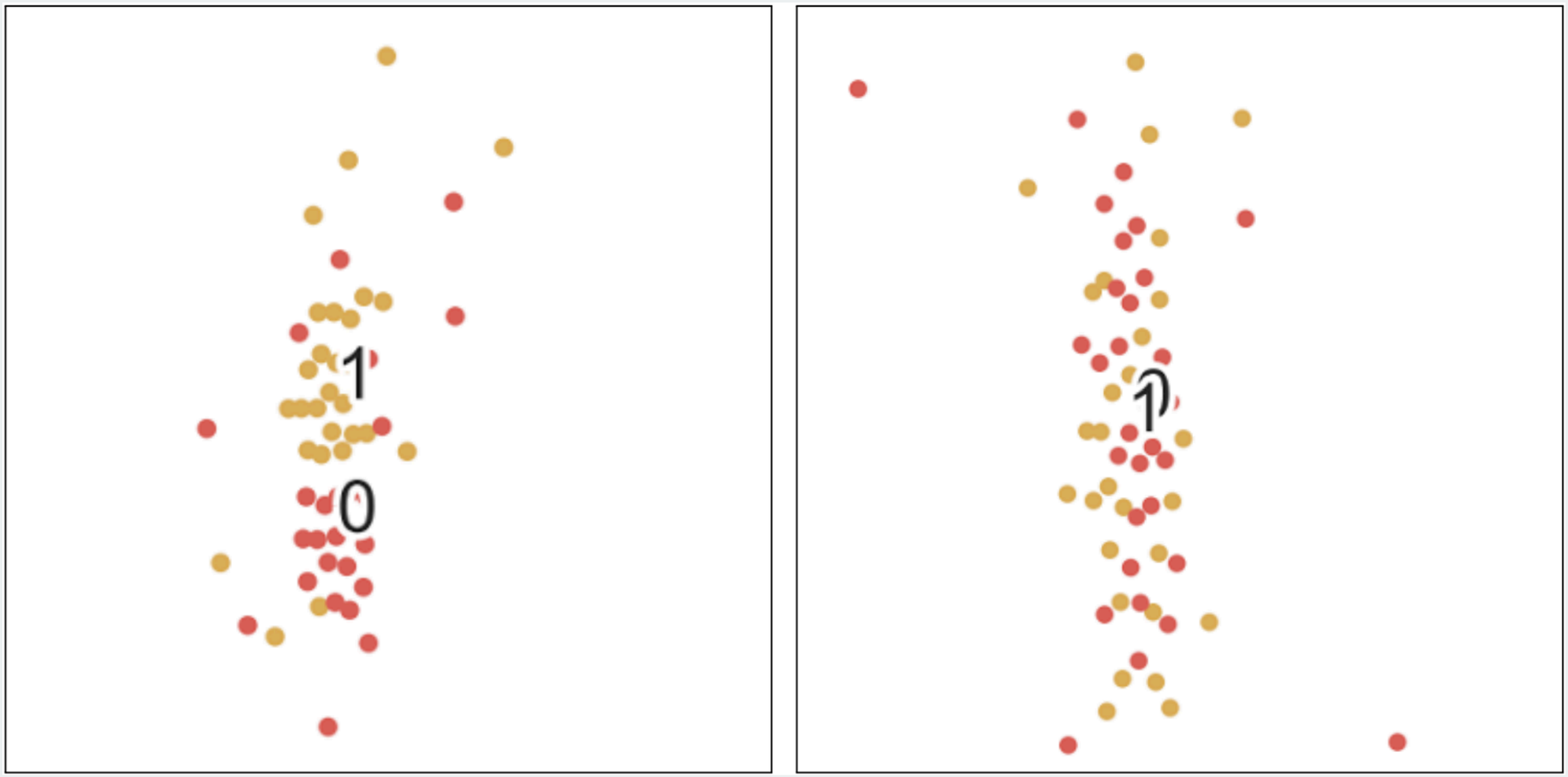
1. Учителю виднее!
2. **Профессор против Учителя**
3. SeqGAN
4. Gumbel-softmax trick
5. Grammar VAE

Professor Forcing



Professor Forcing

Если до этого мы «дискриминировали» сгенерированные объекты, то здесь мы сравниваем между собой состояния в процессе генерации.



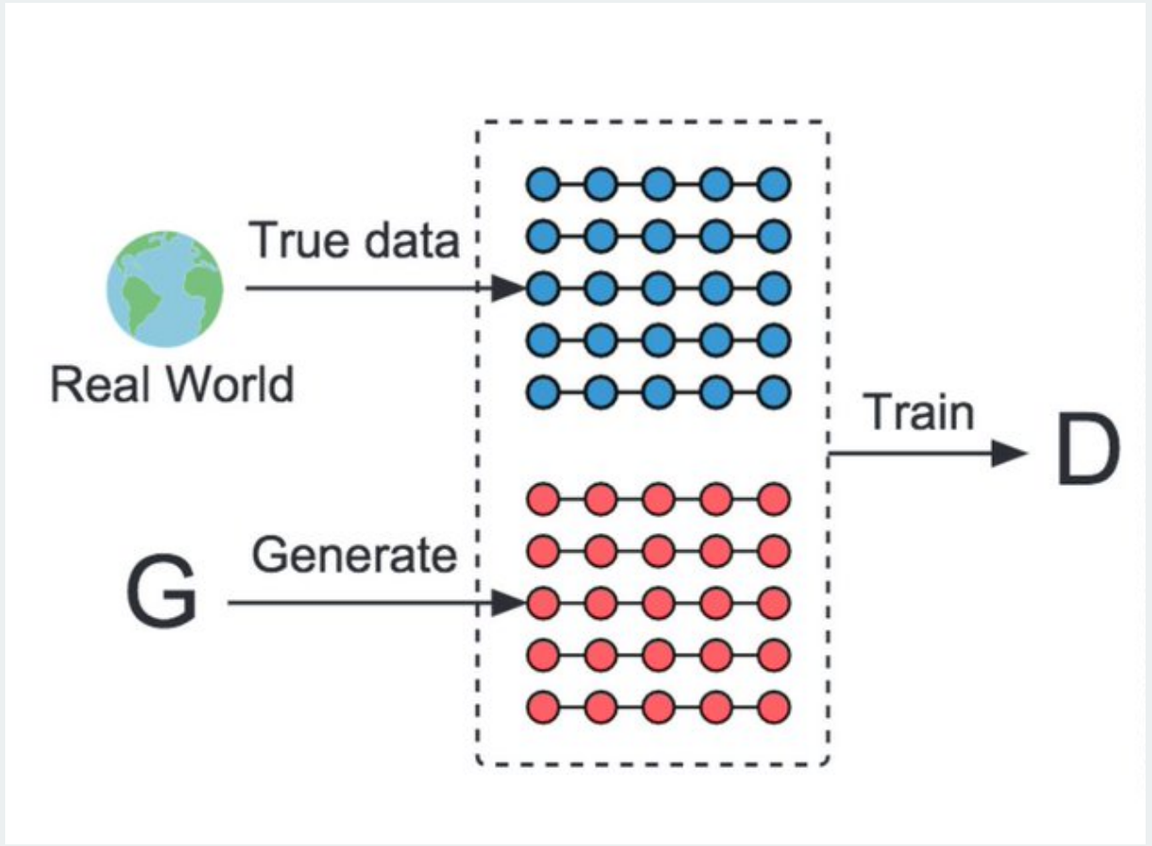
Lamb et al. Professor Forcing: A New Algorithm for Training Recurrent Networks

План на сегодня

1. Учителю виднее!
2. Профессор против Учителя
3. **SeqGAN**
4. Gumbel-softmax trick
5. Grammar VAE

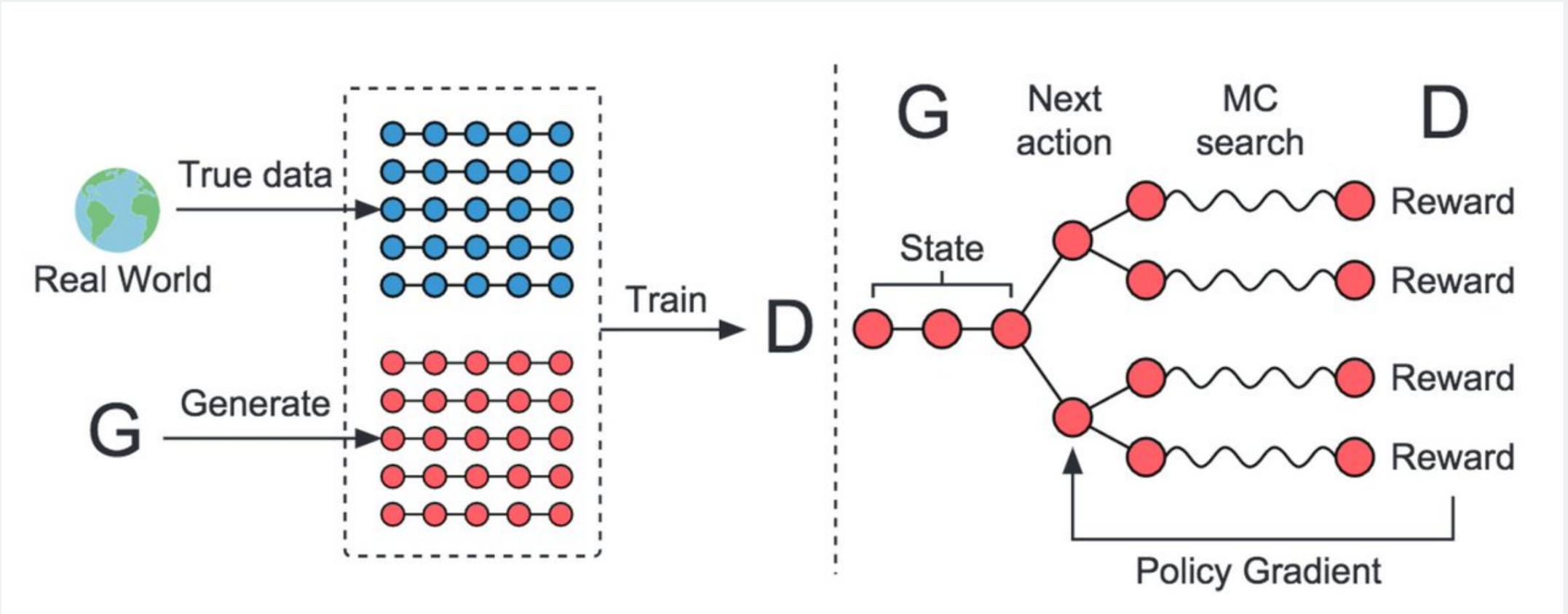
GAN

Почему нельзя просто так взять и начать генерировать текст с помощью состязательной сети?



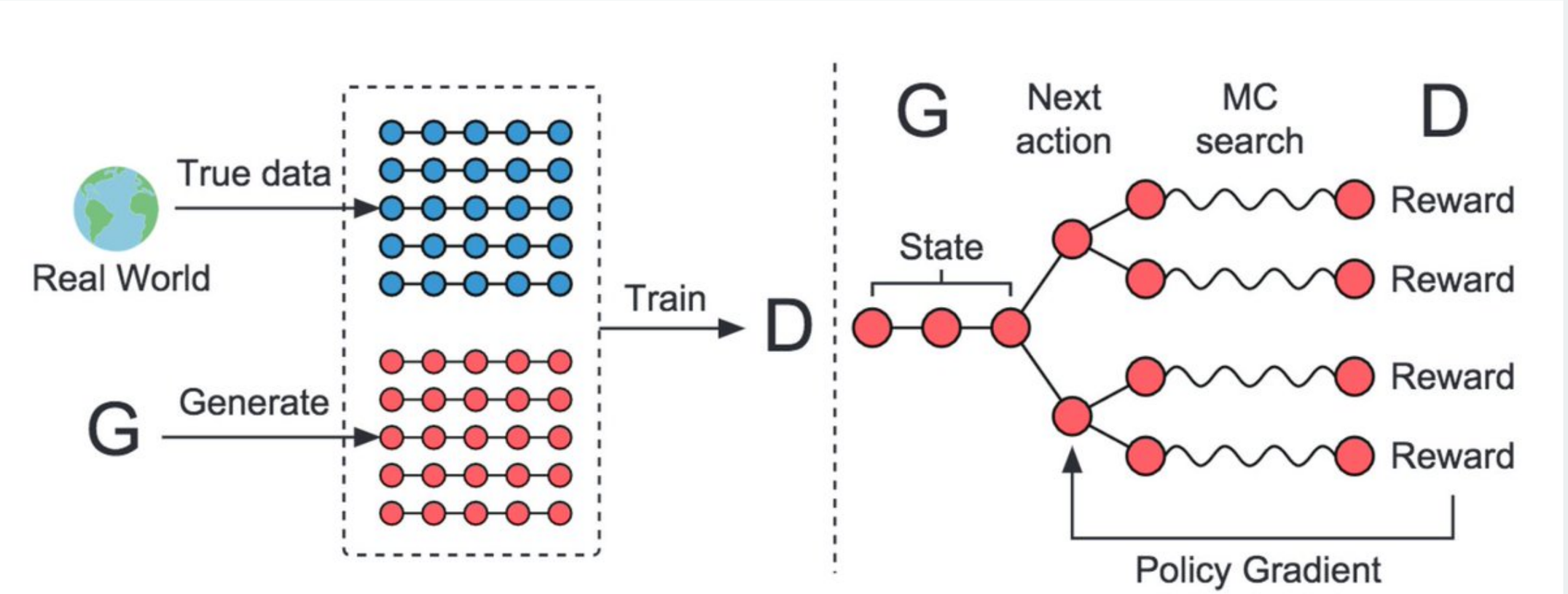
Yu et al. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient

SeqGAN



Так как мы не можем пропустить градиент напрямую сквозь дискретную последовательность, мы используем RL!

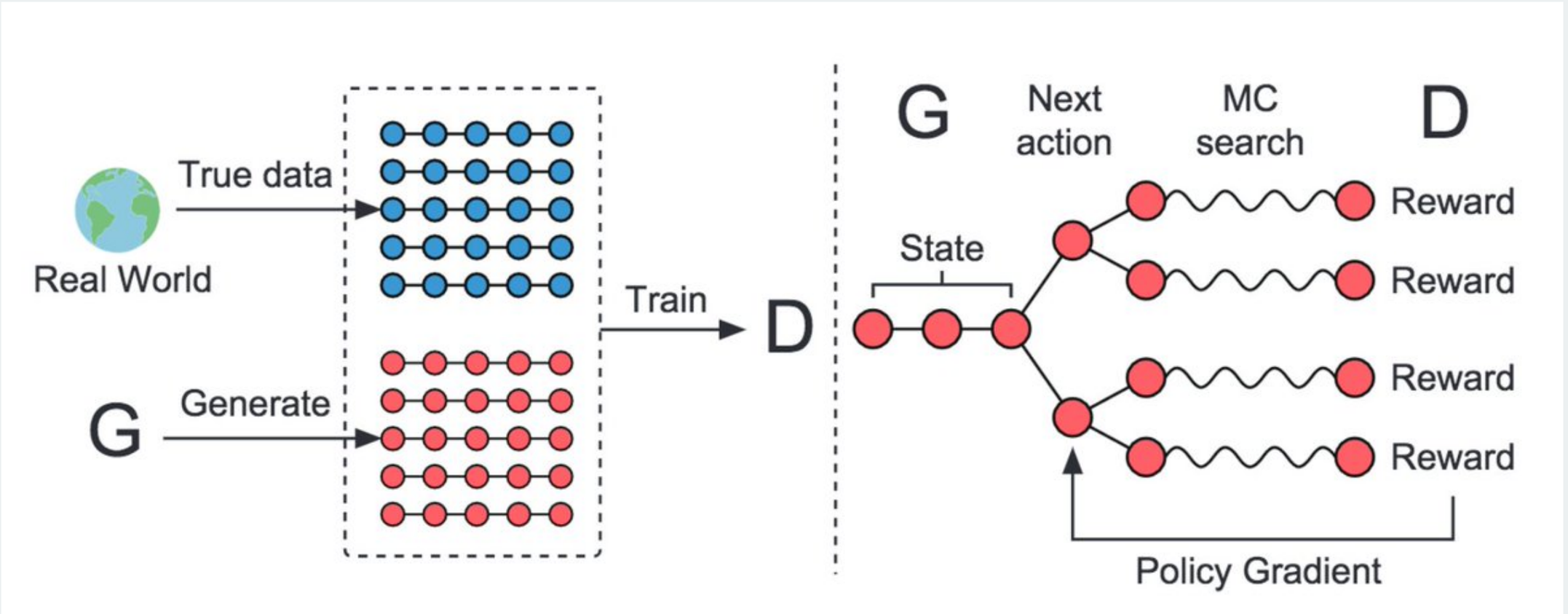
SeqGAN



Для очередного шага генерации мы оцениваем «эффективность» каждого из возможных «действий» и используем ее как ошибку.

Генератор получает обратную связь как E[<вероятность обмануть D>]

SeqGAN



Если выбор каждой следующей буквы назвать действием, то дойдя до «финала» эпизода, т.е. сгенерировав токен остановки, мы можем получить «награду» от Дискриминатора.
Метод Монте-Карло позволит нам получить среднюю оценку для каждого из возможных в текущем состоянии действий, и мы попробуем максимизировать вероятность пойти туда, где ожидания выше.

On-policy RL

Для набора параметров мы можем посчитать вознаграждение как математическое ожидание от вознаграждения за определенную «траекторию», или последовательность токенов w.r.t. вероятность сгенерировать такую последовательность:

$$J(\theta) = \mathrm{E}\Big[\sum_{t=0}^{H} R(s_t, u_t); \pi_\theta\Big] = \sum_{\tau} P(\tau; \theta) R(\tau)$$

https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146

On-policy RL

Для того, чтобы обучать модель, нам нужно получить градиент от J по параметрам:

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(a_{i,t}\middle|s_{i,t}\right)\right)\left(\sum_{t=1}^{T} r\left(s_{i,t}, a_{i,t}\right)\right)$$

Градиент в середине этой формулы это ни что иное как градиент логарифма правдоподобия. Мы хотим максимизировать вероятности пройти по тем траекториям, которые приведут к большой награде и наоборот.

https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146

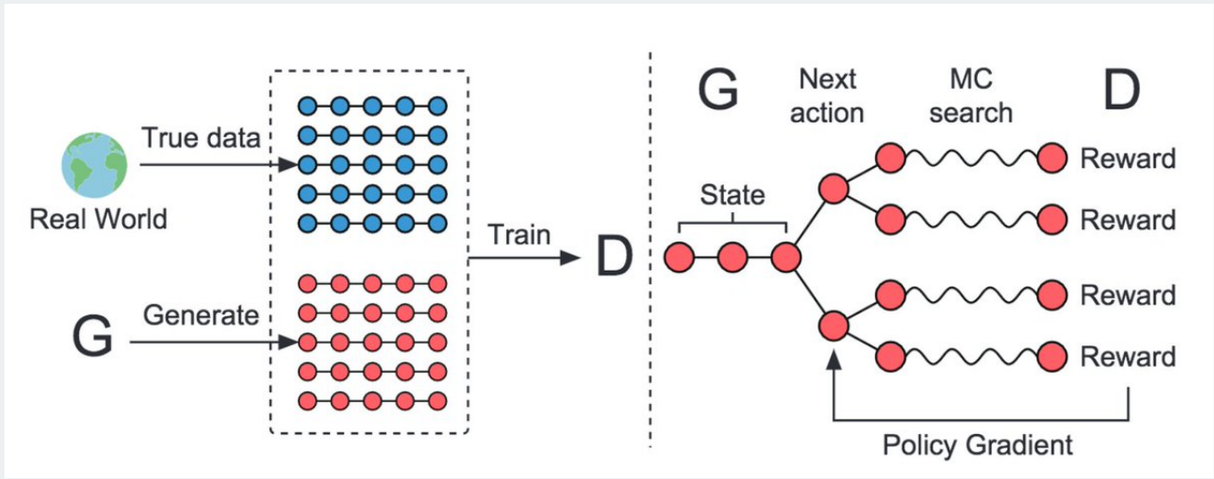
On-policy RL

Мы будем обучать SeqGAN при помощи алгоритма REINFORCE:

REINFORCE algorithm:

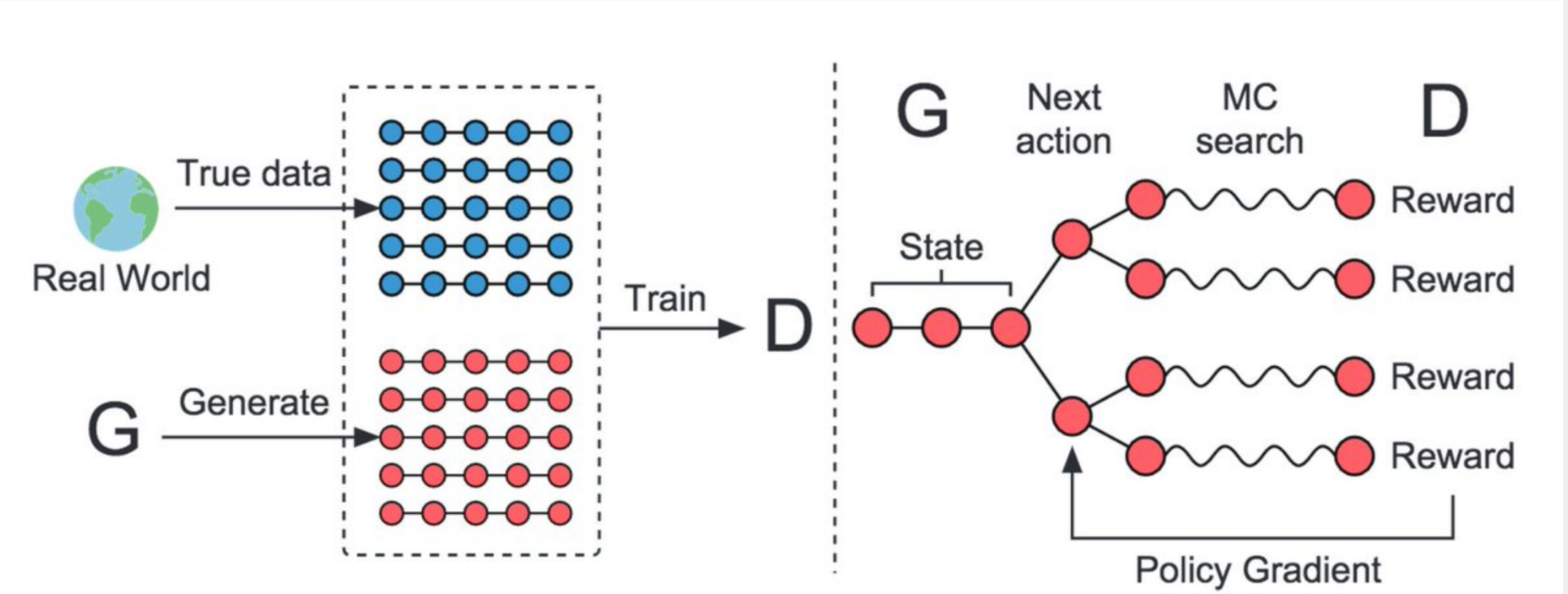
1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Найдем этапы REINFORCE на схеме SeqGAN:



https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146

SeqGAN



Для очередного шага генерации мы оцениваем «эффективность» каждого из возможных «действий» и используем ее как ошибку.

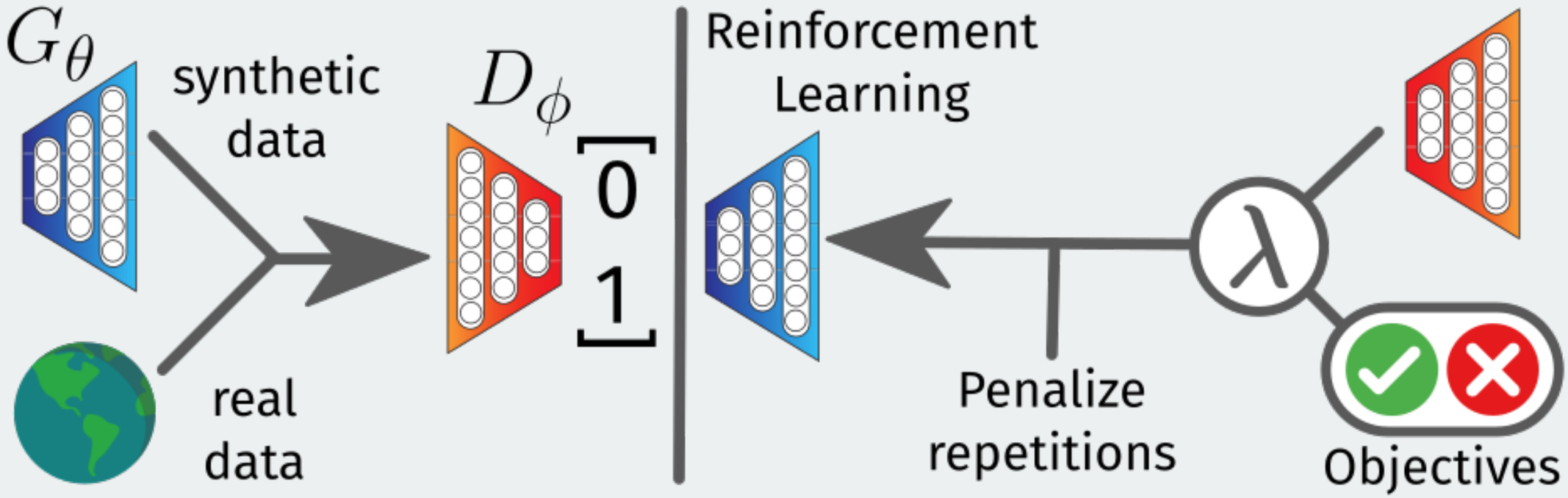
Генератор получает обратную связь как E[<вероятность обмануть D>]

SeqGAN

Реализации: https://github.com/topics/seqgan

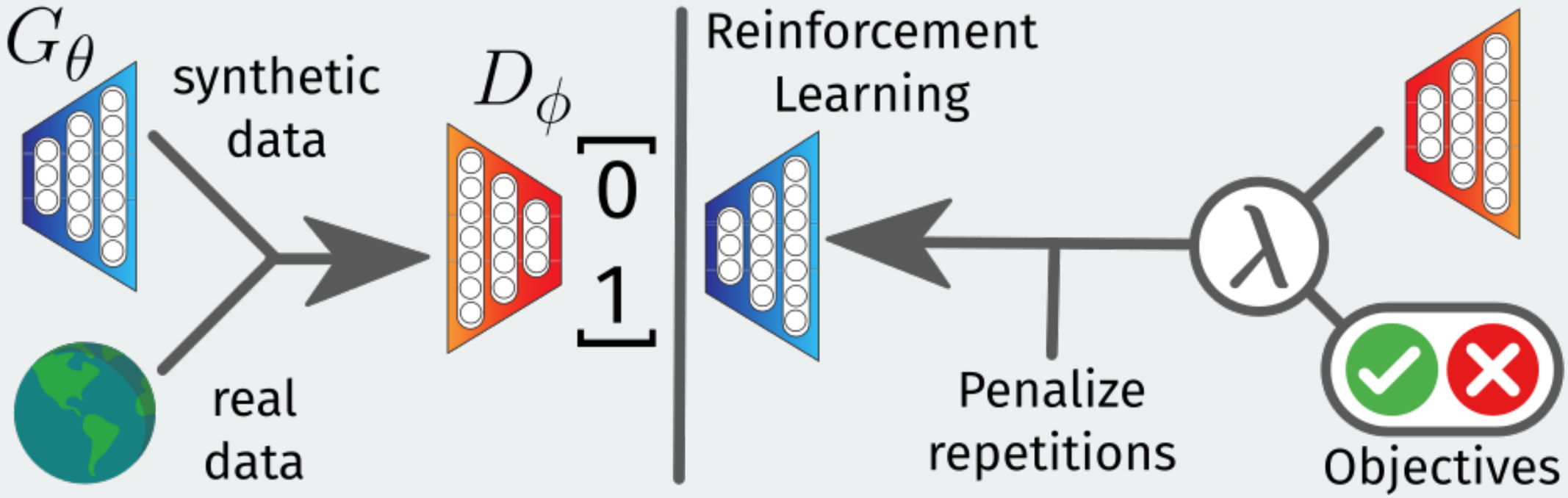
Авторская реализация: https://github.com/LantaoYu/SeqGAN

Objective-Reinforced GANs (ORGAN)



Раз уж мы считаем ошибку как мат.ожидание награды по free run’ам, то почему бы не добавить в нее дополнительных не дифференцируемых слагаемых?

Objective-Reinforced GANs



В качестве Objectives могут выступать оценки от стороннего софта или ассесоров, валидность или разнообразие объектов в сгенерированном батче и т.д.

План на сегодня

1. Учителю виднее!
2. Профессор против Учителя
3. SeqGAN
4. **Gumbel-softmax trick**
5. Grammar VAE

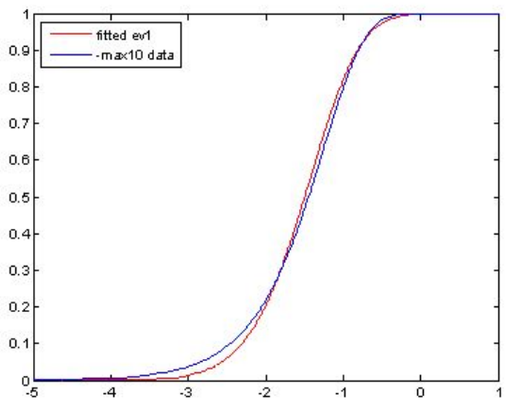
Gumbel-trick

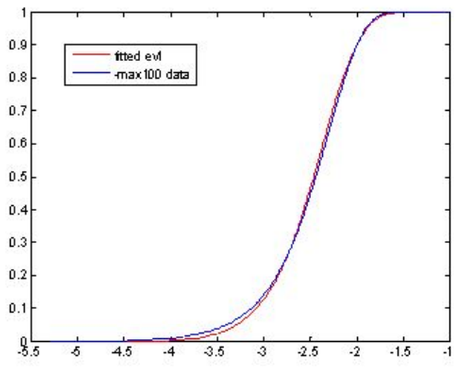
Gumbel distribution

- . $PDF = \frac{1}{\beta}\exp\left(-z - e^{-z}\right), \qquad z = \frac{x-\mu}{\beta} \qquad CDF = \exp(-e^{-z})$
- Mean, median, mode and variance

$Mean = \mu + \beta\gamma \qquad median = \mu - \beta\ln(\ln(2)) \qquad \text{mode} = \mu$

$Variance = \frac{\pi^2}{6}\beta^2 \qquad$ Euler-Mascheroni constant $\gamma \approx 0.5772$





Распределение Гумбеля используется для оценки вероятности того, что максимальная статистика из выборки достигнет определенного максимума

Это полезно при прогнозировании «экстремальных» событий

https://slideplayer.com/slide/6411959/

Gumbel-trick

Reparameterization trick

Как мы поступаем, если в нейросети узлы или слои - случайные?

Gumbel-trick

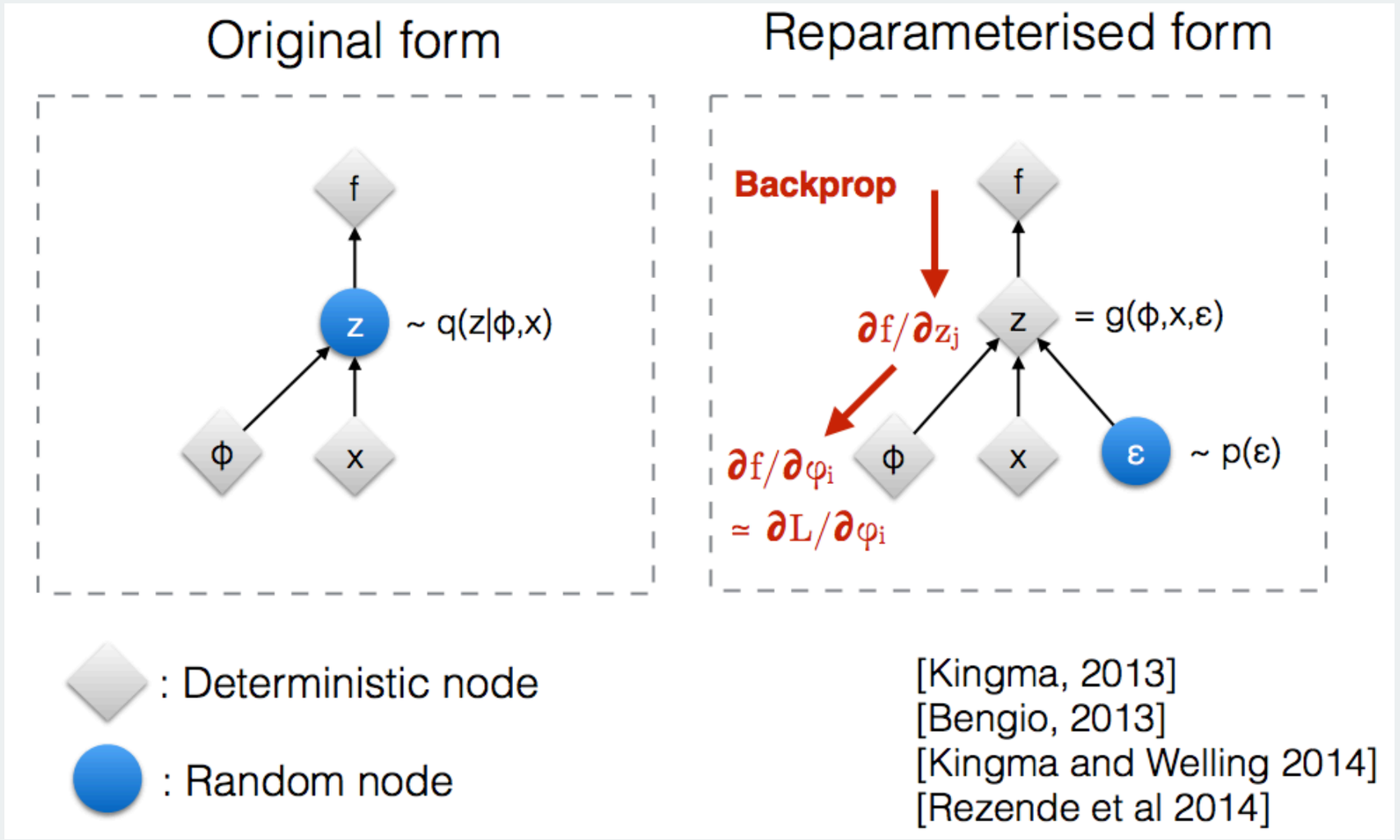
Reparameterization trick

Как мы поступаем, если в нейросети узлы или слои - случайные?

> Мы не можем вычислять градиент на этих узлах
> Поэтому вычисляем градиент для μ, Σ распределения в этом узле (или другие параметры)

Gumbel-trick

Reparameterization trick



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Gumbel-trick

Reparameterization trick

Непрерывное распределение: ✓
Дискретное распределение: ???

Gumbel-trick

Reparameterization trick

Непрерывное распределение: ✓
Дискретное распределение: ???

Почему нельзя использовать reparametrization trick с дискретным распределением?

Gumbel-trick

Reparameterization trick

Непрерывное распределение: ✓
Дискретное распределение: ???

Почему нельзя использовать reparametrization trick с дискретным распределением?

Нельзя получить (*нетривиальную*) дифференцируемую функцию, которая отображает непрерывное множество в дискретное! Следовательно, мы не сможем считать градиент для reparametrization trick’a

Gumbel-trick

Замечательное свойство распределения Гумбеля

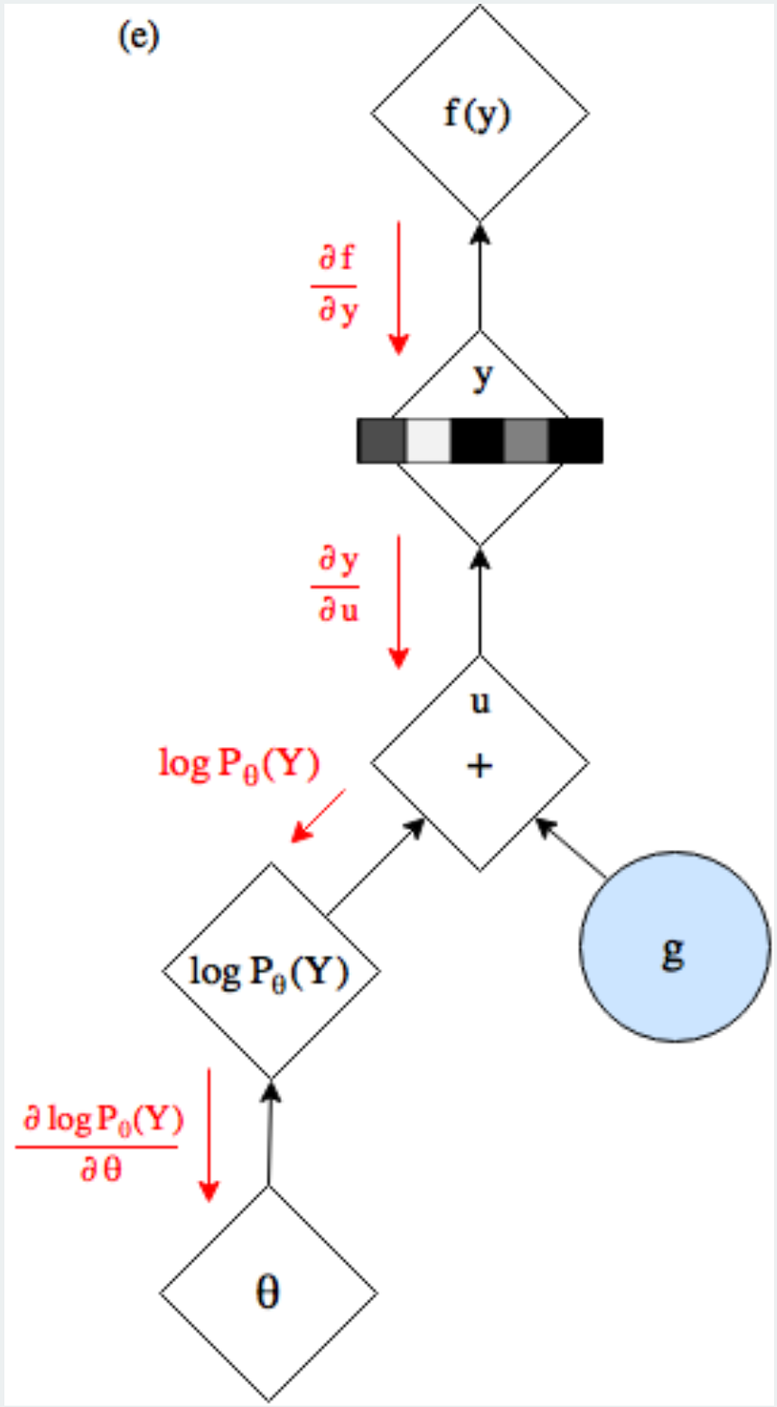
> Вход: дискретное распределение a_i с заданными вероятностями классов i (активации какого-то слоя)
Сгенерируем семплы из распределения так:

$z = one\text{-}hot\ (arg\ max_i\ [g_i + log\ (a_i)\ ])$
$g_i = -log(-log(Uniform(0,1))$

Если сложить значения, взятые из стандартного распределения Гумбеля, и логиты (значения узла, прогнозирующие вероятность i-го класса), а затем вернуть индекс максимального значения, мы получим значение, распределенное как наше оригинальное

Gumbel-trick

Схема трюка



Gumbel-trick

Замечательное свойство распределения Гумбеля

$z = one\text{-}hot\ (arg\ max_i\ [g_i + log\ (a_i)\ ])$

Проблемы?

Gumbel-trick

Замечательное свойство распределения Гумбеля

$z = one\text{-}hot\ (arg\ max_i\ [g_i + log\ (a_i)\ ])$

Проблемы?

> arg-max недифференцируем

Решения?

Gumbel-trick

Замечательное свойство распределения Гумбеля

$z = one\text{-}hot\ (arg\ max_i\ [g_i + log\ (a_i)\ ])$

Проблемы?

> arg-max недифференцируем

Решения?

Использовать softmax для того, чтобы приближенно моделировать arg-max + one-hot.

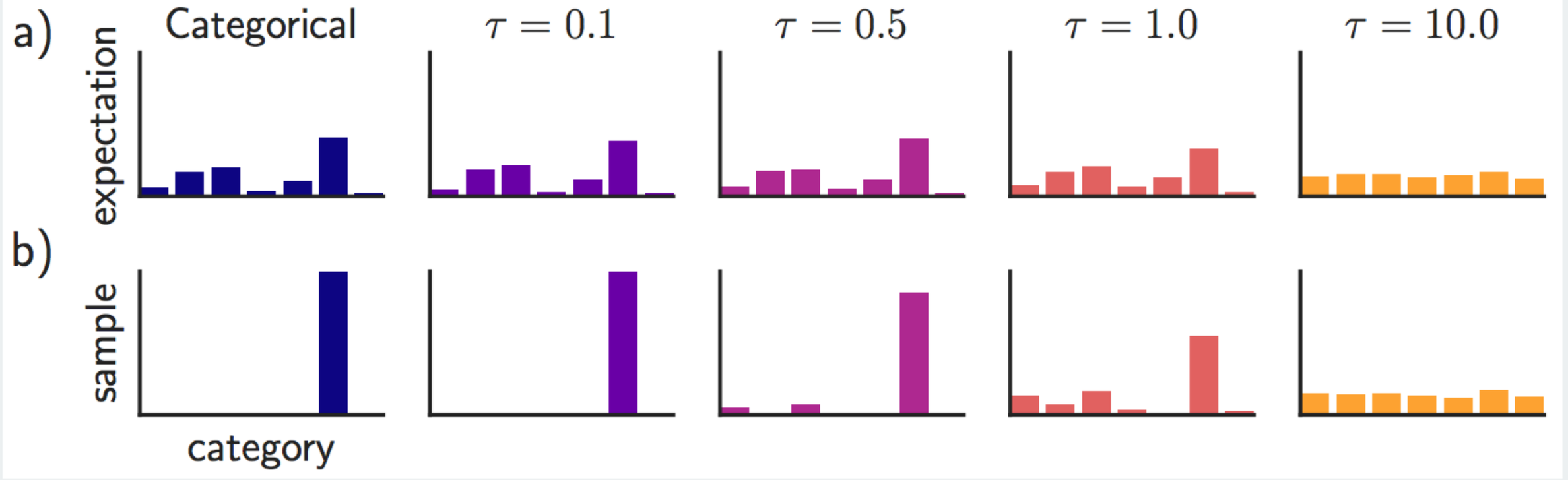
Подробности в коде!

Gumbel Softmax

$$\boldsymbol{y} = softmax\left(\frac{\boldsymbol{h} + \boldsymbol{g}}{\tau}\right)$$

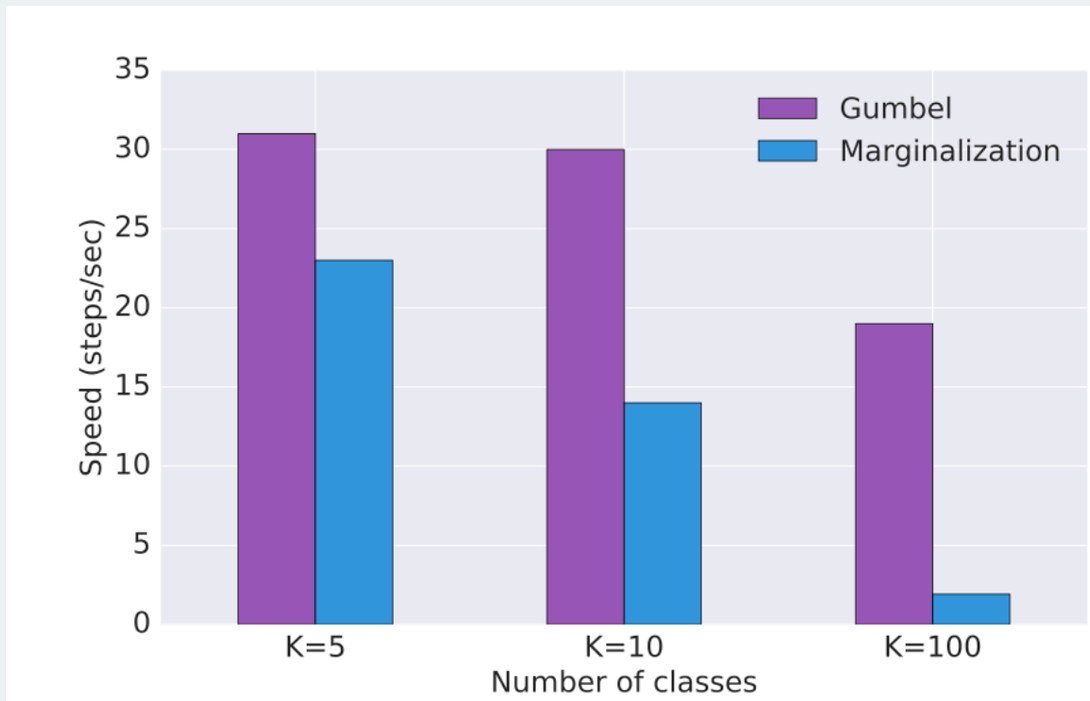
Где $\boldsymbol{h}$ — это логиты, $\boldsymbol{g}$ — это сэмпл из распределения Гумбеля и τ — обратная температура.

Если τ стремится к 0, то это распределение вырождается в argmax!

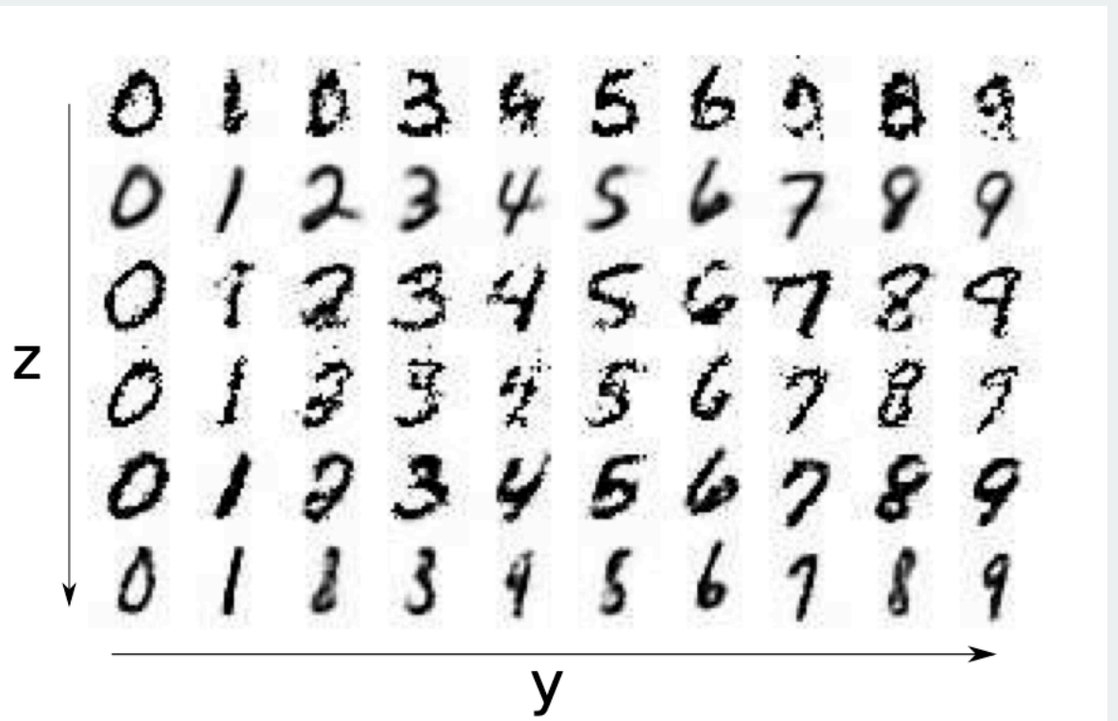


Зачем это все?

Semi-supervised learning с использованием Gumbel Trick проходит значительно быстрее. И чем больше классов, тем быстрее работает метод.



(a)



(b)

Grammar VAE

Одна из больших проблем при генерации данных со сложной структурой заключается в том, что очень сложно сделать генерацию и разнообразной и валидной одновременно.
Если мы генерируем предложение по словам, то окажется ли оно валидным или зависит от того научилась модель «валидности» или нет, но добиться этого в общем случае очень сложно.
Почему?

Grammar VAE

Одна из больших проблем при генерации данных со сложной структурой заключается в том, что очень сложно сделать генерацию и разнообразной и валидной одновременно.
Если мы генерируем предложение по словам, то окажется ли оно валидным или зависит от того научилась модель «валидности» или нет, но добиться этого в общем случае очень сложно.
А что если генерацию подчинить правилам?

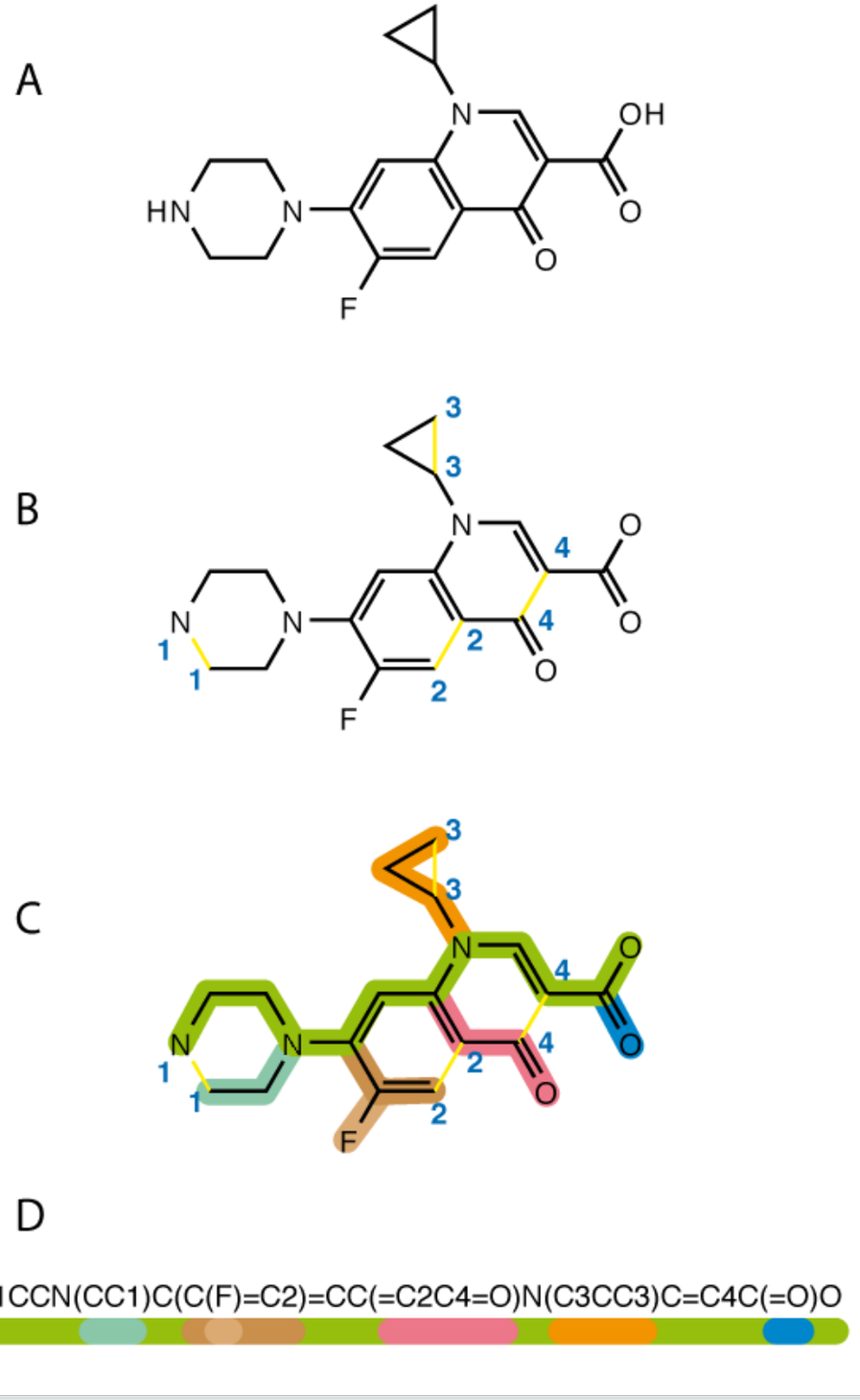
План на сегодня

1. Учителю виднее!
2. Профессор против Учителя
3. SeqGAN
4. Gumbel-softmax trick
5. **Grammar VAE**

Генерация молекул

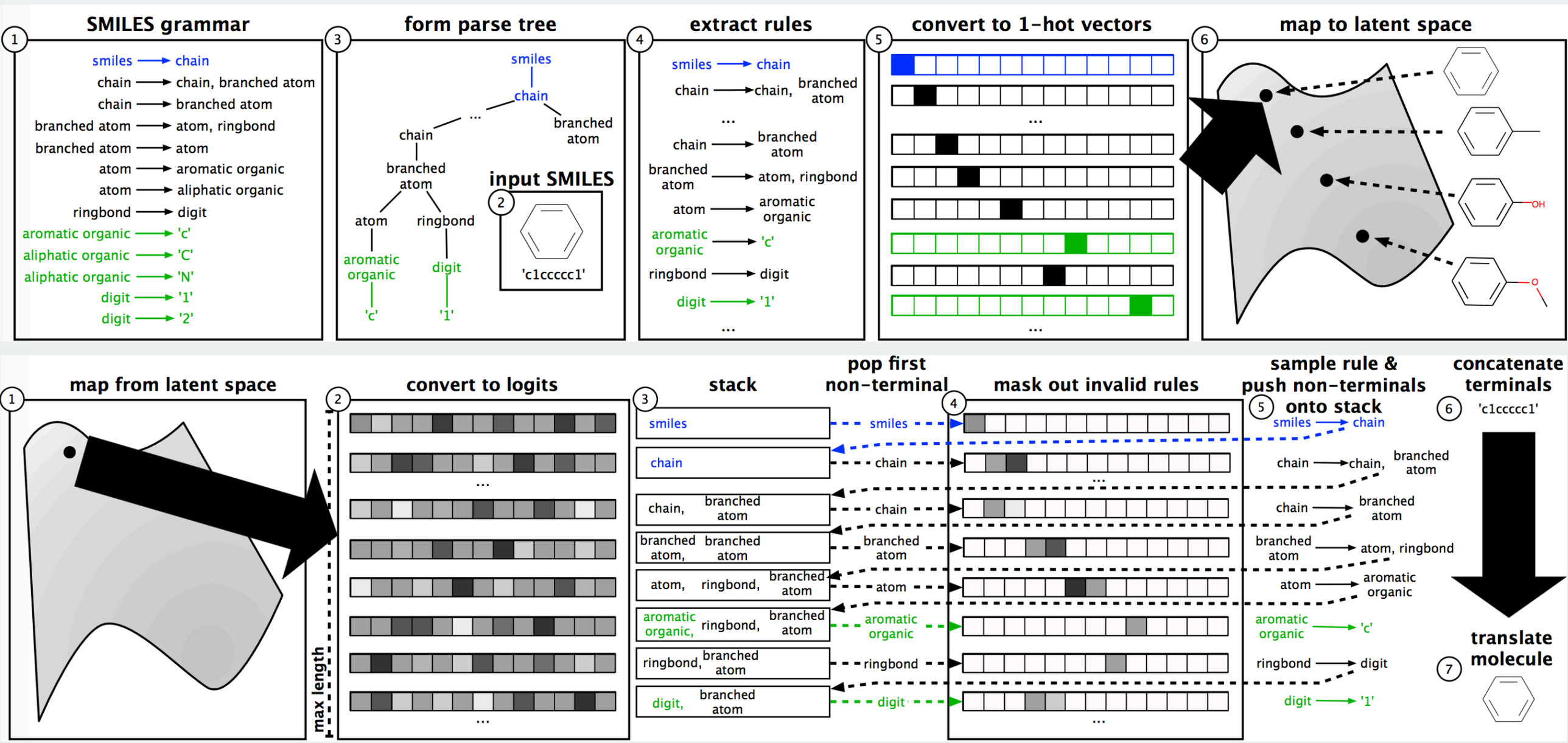
Молекулу, с потерей пространственной информации, можно представить в виде графа. Любой граф можно рекурсивно обойти и записать в строку. Самой большой проблемой являются циклы, из-за них приходится делать разрезы и помечать ребра идентификаторами.

С одной стороны, запись в строку можно использовать для обучения моделей как есть, с другой, у такого представления есть довольно сложная неочевидная структура.

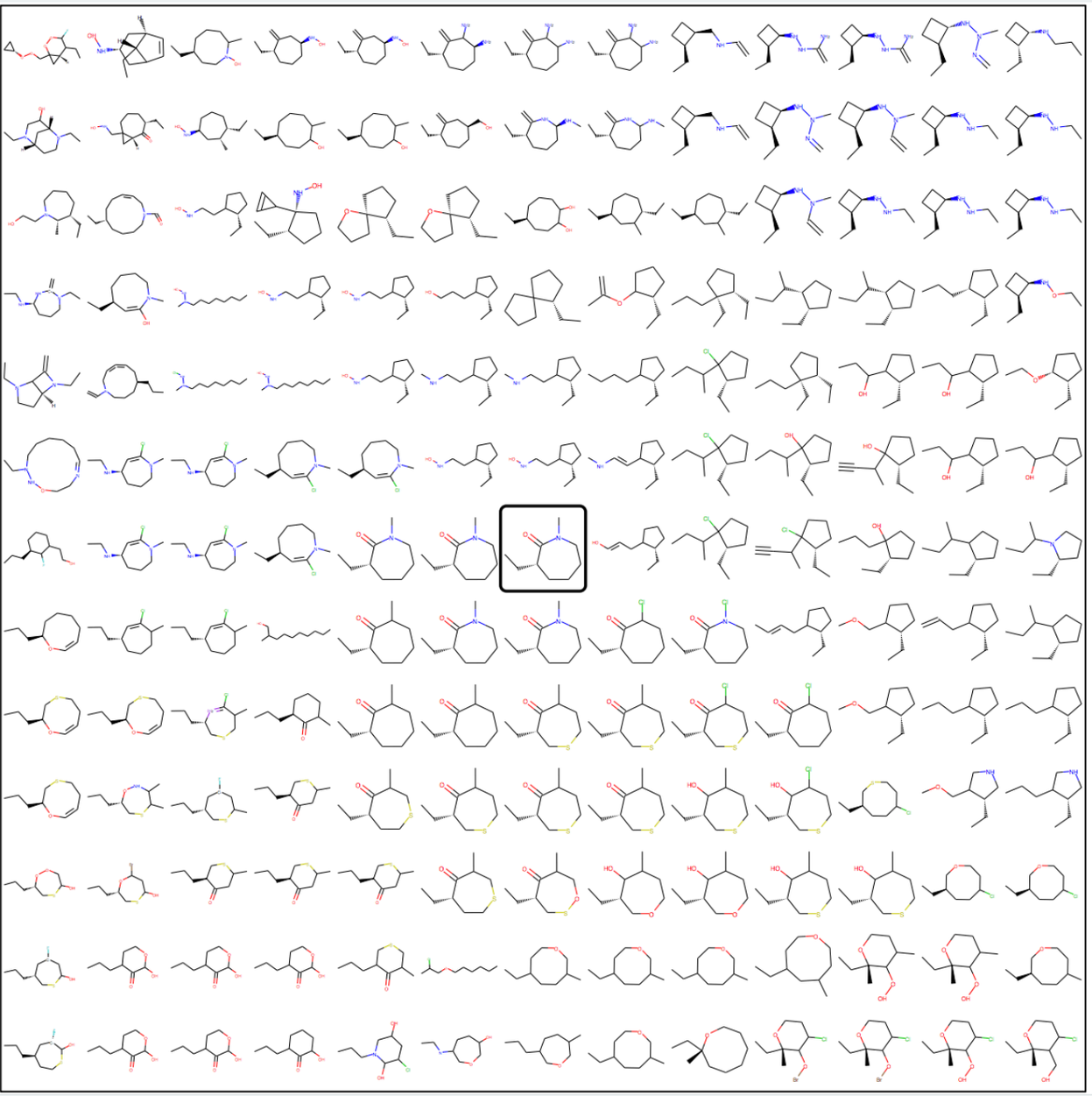


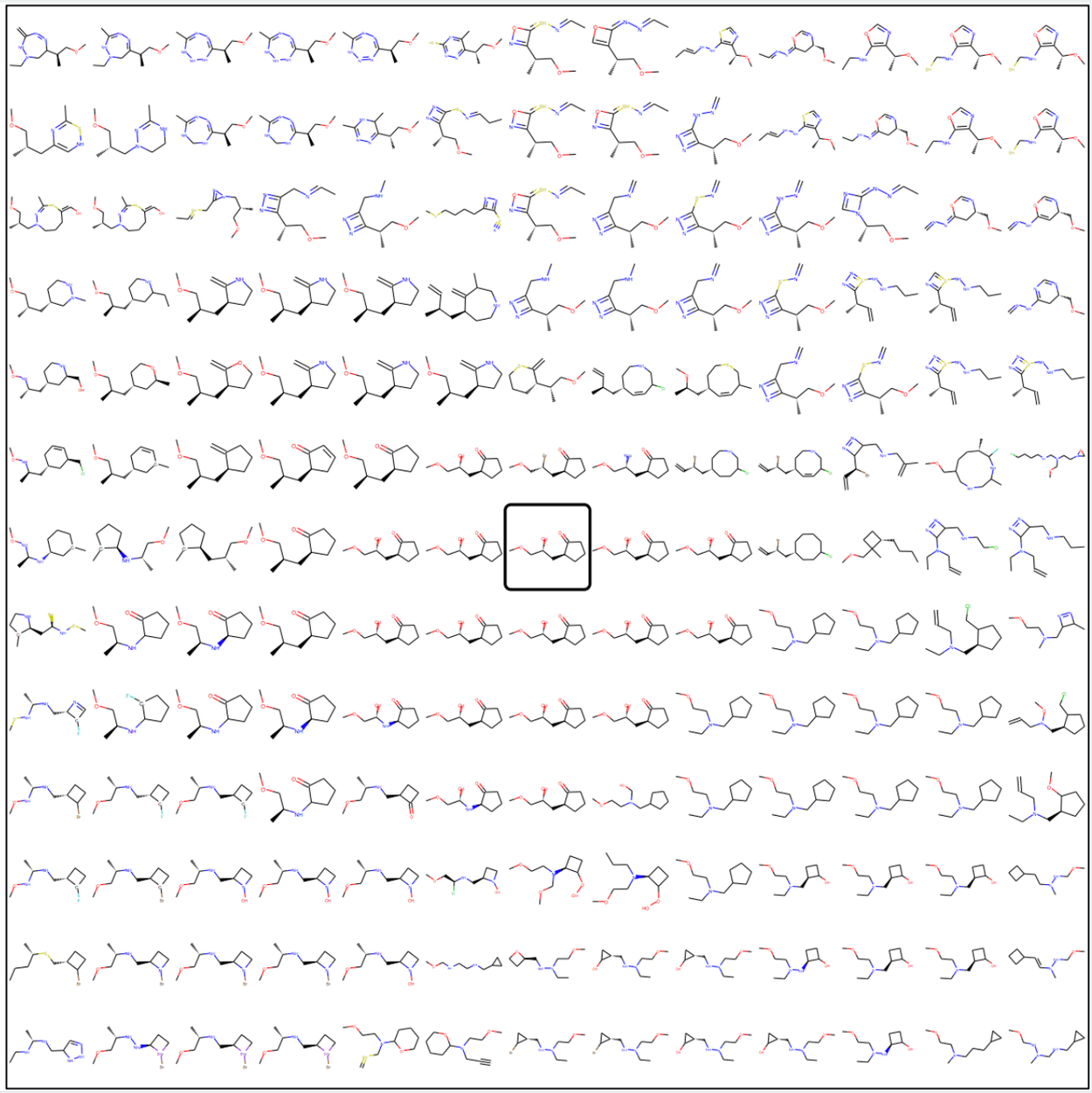
Grammar VAE

Если мы можем описать объект в виде цепочки применения правил, то какую бы цепочку мы не сгенерировали, мы получим валидный объект



Grammar VAE





Спасибо
за внимание!