



Общий искусственный интеллект: подходы и нерешенные проблемы

Алексей Потапов

Семинар Сибирской школы искусственного интеллекта, 24 декабря 2019

Подходы к AGI

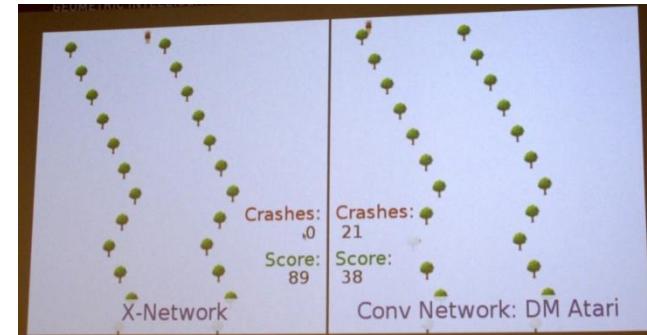
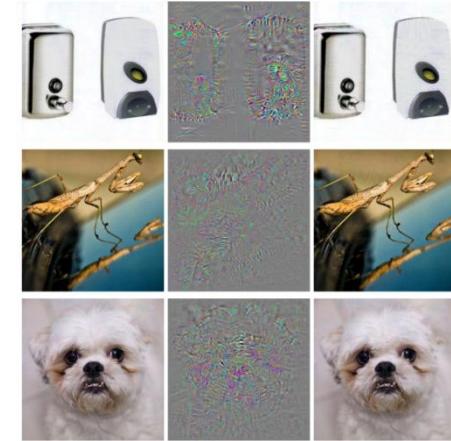
- Глубокое обучение
- Когнитивные архитектуры
- Вероятностные модели
- Универсальный алгоритмический интеллект
- Обучение с подкреплением
- ...

Глубокое обучение: польза



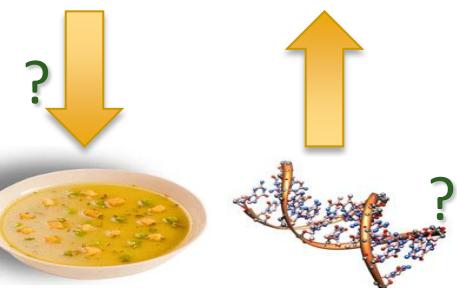
Критика глубокого обучения

- Слабое обобщение
 - Необходимы большие выборки; нет обучения по одному примеру
 - Невозможность обучения инвариантам
 - Наличие уязвимостей
 - Проблемы с трансферным обучением и обучением без учителя
- А также
 - Кодирует статистики, но не логические, каузальные или структурные отношения
 - Трудности в рассуждениях и планировании

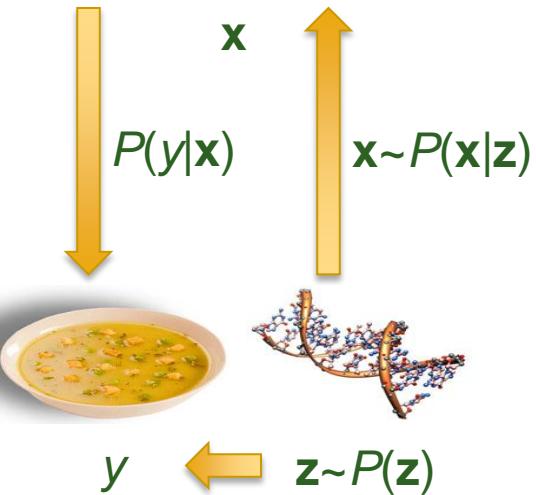


Изображения из: Szegedy, C. et al. Intriguing properties of neural networks. arXiv 1312.6199 (2013).
Gary Marcus. Keynote @ AGI-16

Генеративные и дискриминантные модели



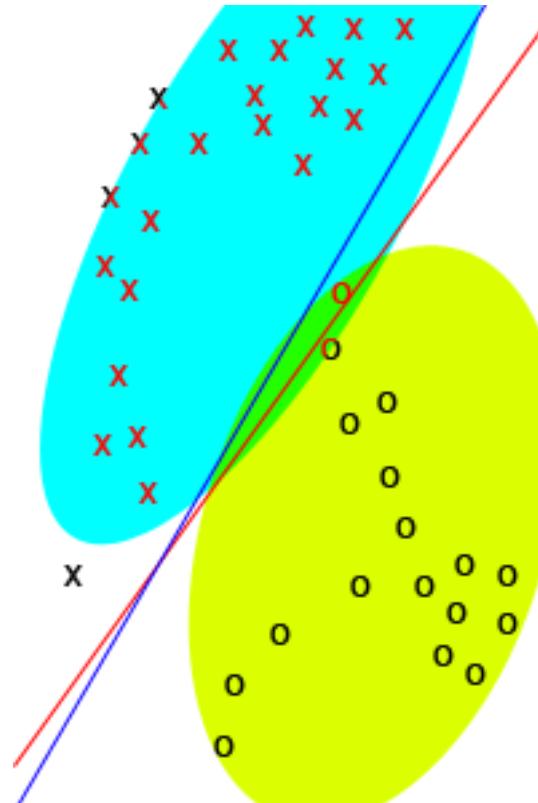
Добавим вероятностей



Вероятностные модели

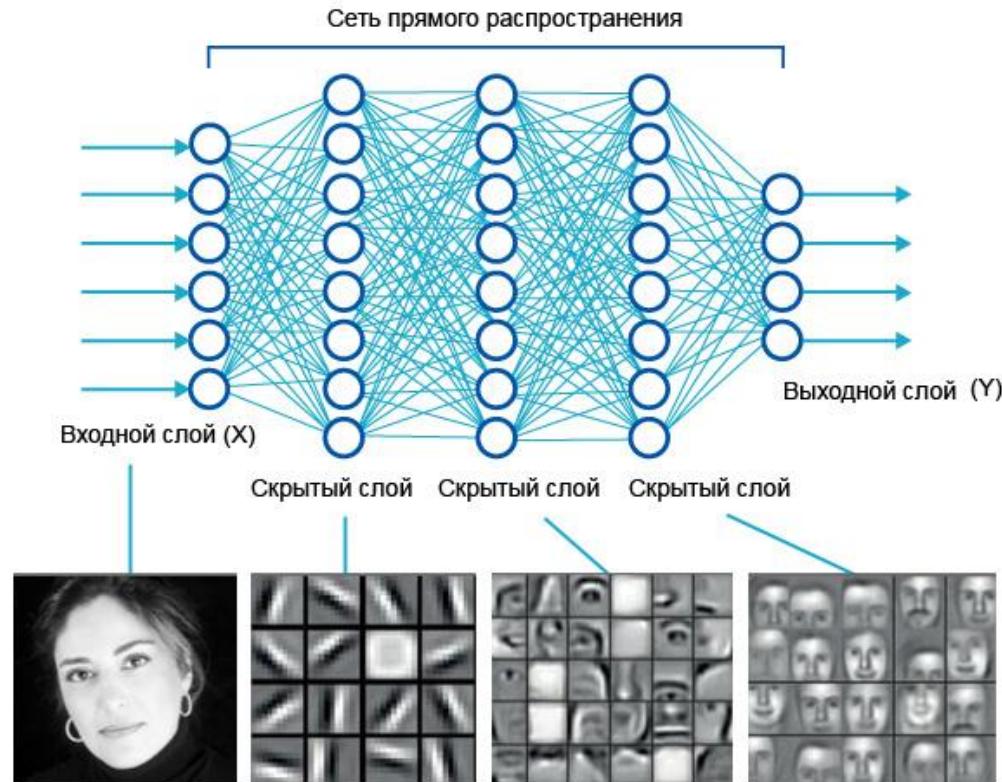
	Дискриминантные	Генеративные
	Отображение из пространства данных в пространство решений $P(y x)$	Отображение из пространства решений в пространство данных $z \sim P(z), x \sim P(x z)$
Pros	<ul style="list-style-type: none">• Эффективные• Меньше предположений о распределении данных	<ul style="list-style-type: none">• Гибкие• Обучение без учителя и с частично размеченными данными
Cons	<ul style="list-style-type: none">• Вывод только в одну сторону• Только обучение с учителем	<ul style="list-style-type: none">• Дополнительные предположения о распределении данных• Вывод вычислительно неэффективен

Вероятностные модели

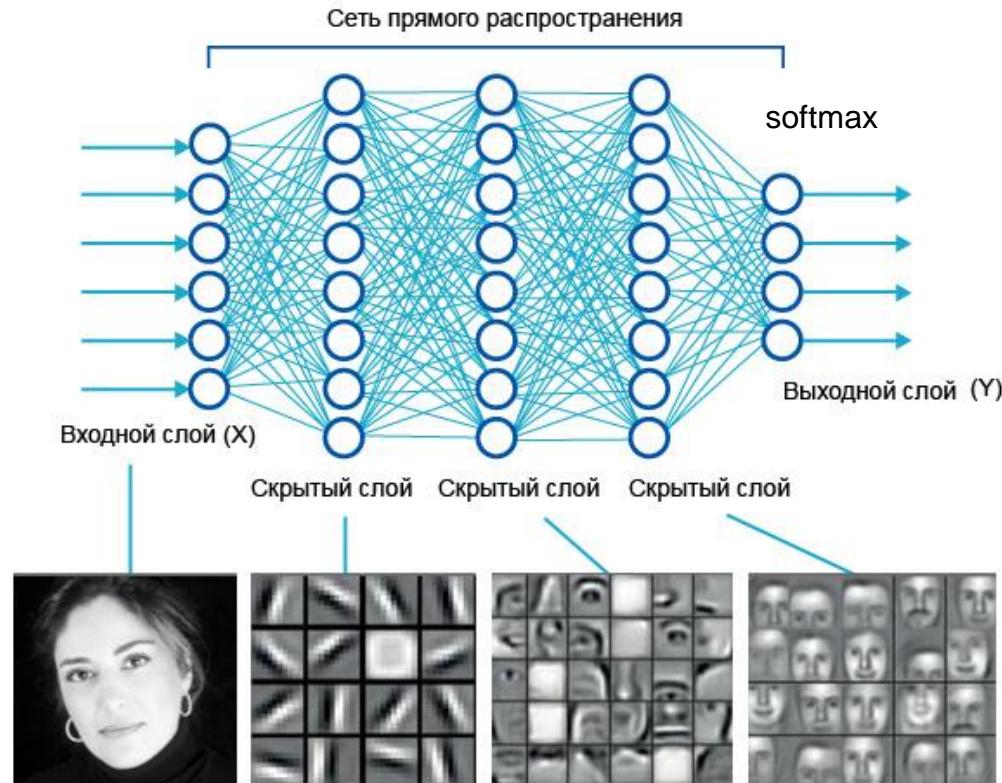


генеративная
дискриминантная

MLP как дискриминантные модели



MLP как дискриминантные модели



Все вместе – прямая аппроксимация $P(y|x)$

Генеративные модели

- Порождение x по z
- В обратную сторону?

$$P(\mathbf{z}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{P(\mathbf{x})} = \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{\sum_{\mathbf{z}} P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}$$

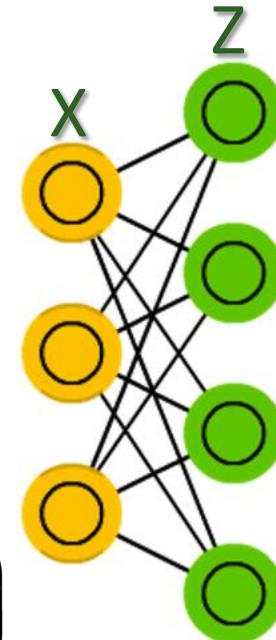
Генеративные модели: RBM

- Порождение x по z
- В обратную сторону?

$$P(\mathbf{z}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{P(\mathbf{x})} = \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{\sum_{\mathbf{z}} P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}$$

- Простейшая нетривиальная вероятностная графическая модель

$$P(\mathbf{x}) = \sum_{\mathbf{z}} P(\mathbf{x}, \mathbf{z}) = \frac{1}{Z} \sum_{\mathbf{z}} e^{\mathbf{b}\mathbf{x} + \mathbf{c}\mathbf{z} + \mathbf{x}^T W \mathbf{z}} \quad P(z_i = 1 | \mathbf{x}) = \sigma\left(\sum_j w_{ij} x_j + c_i\right)$$



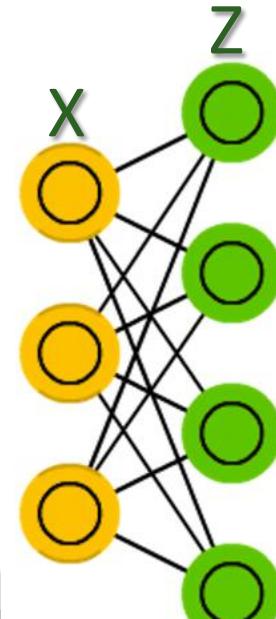
Генеративные модели: RBM

- Порождение x по z
- В обратную сторону?

$$P(\mathbf{z}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{P(\mathbf{x})} = \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{\sum_{\mathbf{z}} P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}$$

- Простейшая нетривиальная вероятностная графическая модель

$$P(\mathbf{x}) = \sum_{\mathbf{z}} P(\mathbf{x}, \mathbf{z}) = \frac{1}{Z} \sum_{\mathbf{z}} e^{\mathbf{b}\mathbf{x} + \mathbf{c}\mathbf{z} + \mathbf{x}^T W \mathbf{z}} \quad P(z_i = 1 | \mathbf{x}) = \sigma\left(\sum_j w_{ij} x_j + c_i\right)$$

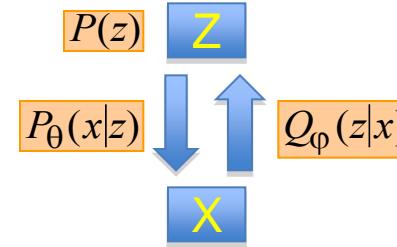


- Та же самая сигмоида, но выучивается $P(x, z)$

Вариационные приближения

- Вычислять апостериорное распределение всегда сложно

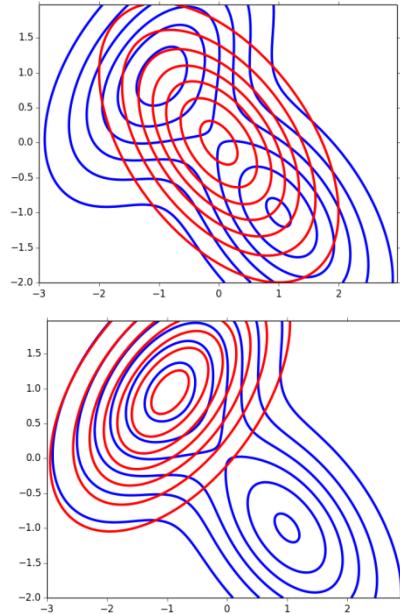
$$P(\mathbf{z}_i | \mathbf{x}_i, \theta) = \frac{P(\mathbf{x}_i, \mathbf{z}_i | \theta)}{P(\mathbf{x}_i | \theta)} = \frac{P(\mathbf{x}_i, \mathbf{z}_i | \theta)}{\int P(\mathbf{x}_i, \mathbf{z} | \theta) d\mathbf{z}}$$



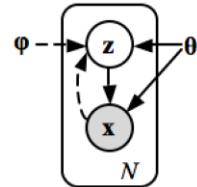
- Будем его аппроксимировать чем-то легко вычислимым $Q(\mathbf{z}|\mathbf{x}_i, \varphi)$
- Критерий: дивергенция Кульбака-Лейблера / вариационная нижняя граница маргинального правдоподобия

$$\log P(\mathbf{x}_i | \theta) = D_{KL}(Q(\mathbf{z}|\mathbf{x}_i, \varphi) \| P(\mathbf{z}|\mathbf{x}_i, \theta)) + L(\theta, \varphi | \mathbf{x}_i)$$

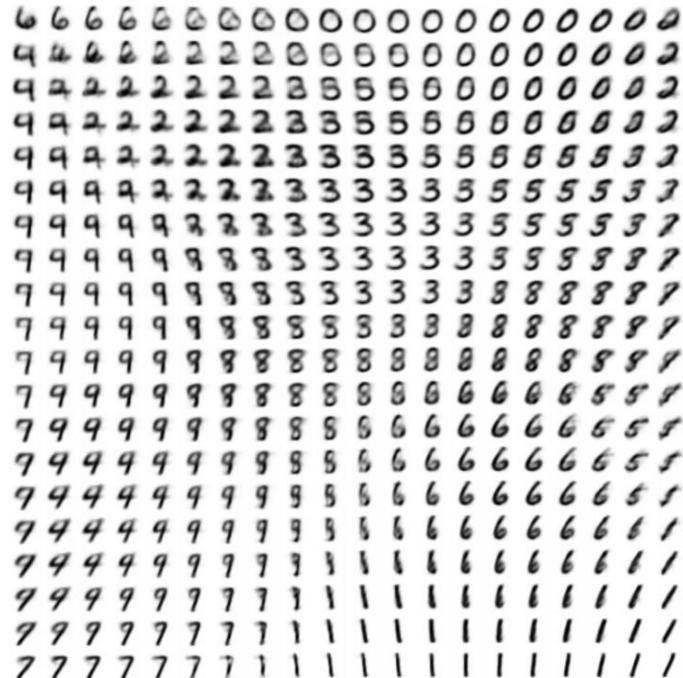
$$D_{KL}(Q \| P) = - \int Q(\mathbf{z}|\mathbf{x}_i, \varphi) \left[\log \frac{P(\mathbf{z}|\mathbf{x}_i, \theta)}{Q(\mathbf{z}|\mathbf{x}_i, \varphi)} \right] d\mathbf{z} \quad L(\theta, \varphi | \mathbf{x}_i) = \int Q(\mathbf{z}|\mathbf{x}_i, \varphi) \left[\log \frac{P(\mathbf{x}_i, \mathbf{z} | \theta)}{Q(\mathbf{z}|\mathbf{x}_i, \varphi)} \right] d\mathbf{z}$$



Вариационные автоэнкодеры



- Давайте учить генеративную модель и ее вариационное приближение одновременно
- Давайте эти распределения представлять как глубокие сети
- Добавим эвристик



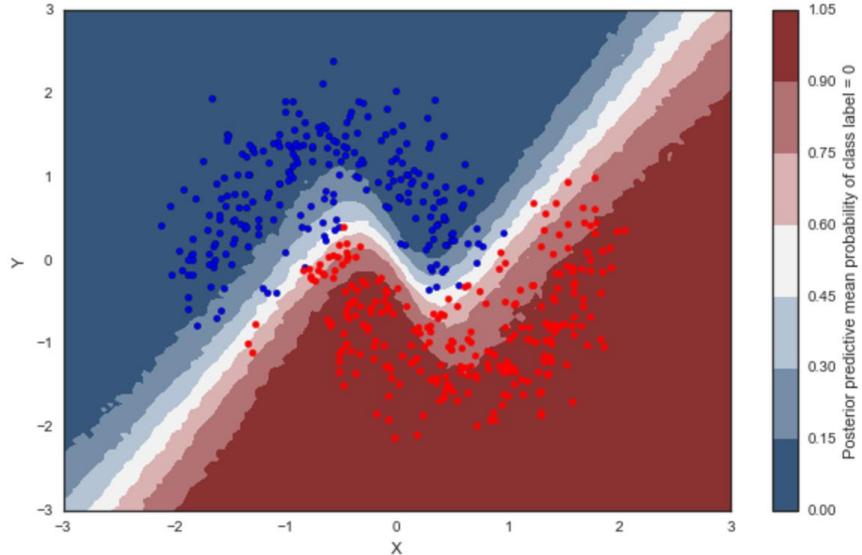
Генеративные конкурирующие сети

- Качество генеративной модели оценивается напрямую дискриминантной моделью, которая пытается отличить образы, порожденные генеративной моделью, от реальных и которая учится вместе с генеративной моделью
- Никакого семплирования; обучение градиентным спуском;
- Множество приложений, особенно, в генерации изображений
- DCGAN, WGAN, LSGAN, ... BiGAN, InfoGAN, Bayesian GAN



Нейробайесовский подход

- Представить распределения вероятностей в виде глубоких сетей
- Ввести распределение вероятностей над весами связей
- Использование приоров
- Оценка неопределенности
- Градиентный спуск + техники байесовского вывода
- Эвристики DL как приближения к Байесовскому выводу



Хорошо ли аппроксимируется $P(x)$?



Известные ориентации

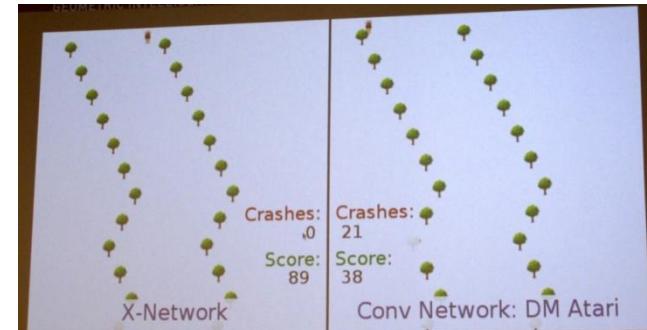
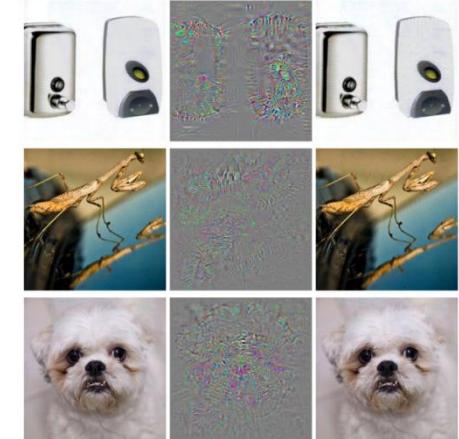


Новые (только для 3 и 4)
ориентации

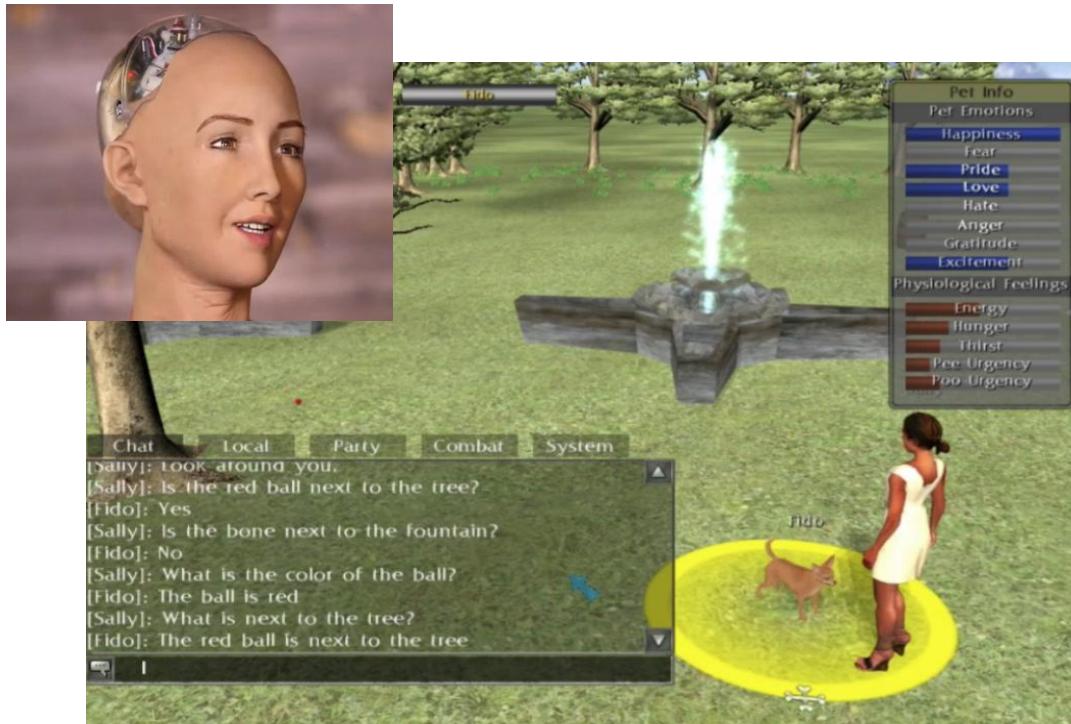
- Многие эксперименты показывают, что глубокие генеративные модели плохо описывают $P(x)$

Критика глубокого обучения

- Слабое обобщение
 - Необходимы большие выборки; нет обучения по одному примеру
 - Невозможность обучения инвариантам
 - Наличие уязвимостей
 - Проблемы с трансферным обучением и обучением без учителя
- А также
 - Кодирует статистики, но не логические, каузальные или структурные отношения
 - Трудности в рассуждениях и планировании



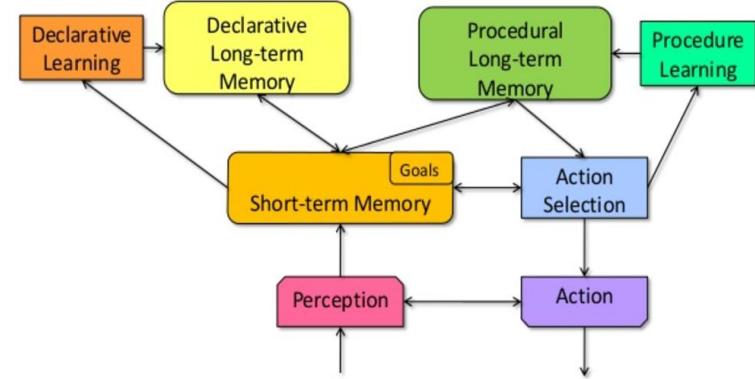
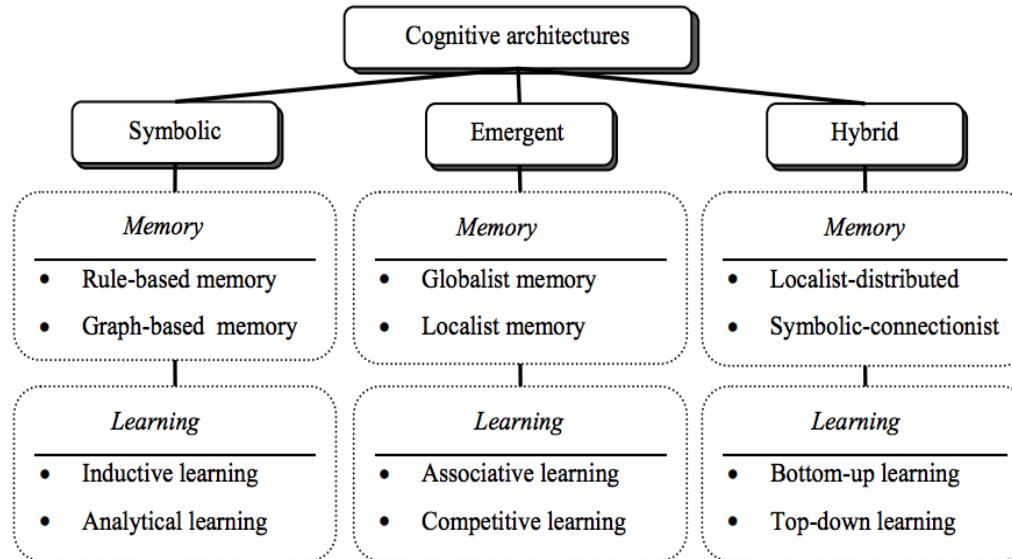
Помогут ли Когнитивные Архитектуры?



- Управление роботами
- Медицинская диагностика
- Обработка естественного языка
- Управление персоналом
- Бионформатика
- ...
- Soar
- LIDA
- OpenCog
- OpenNARS
- ...

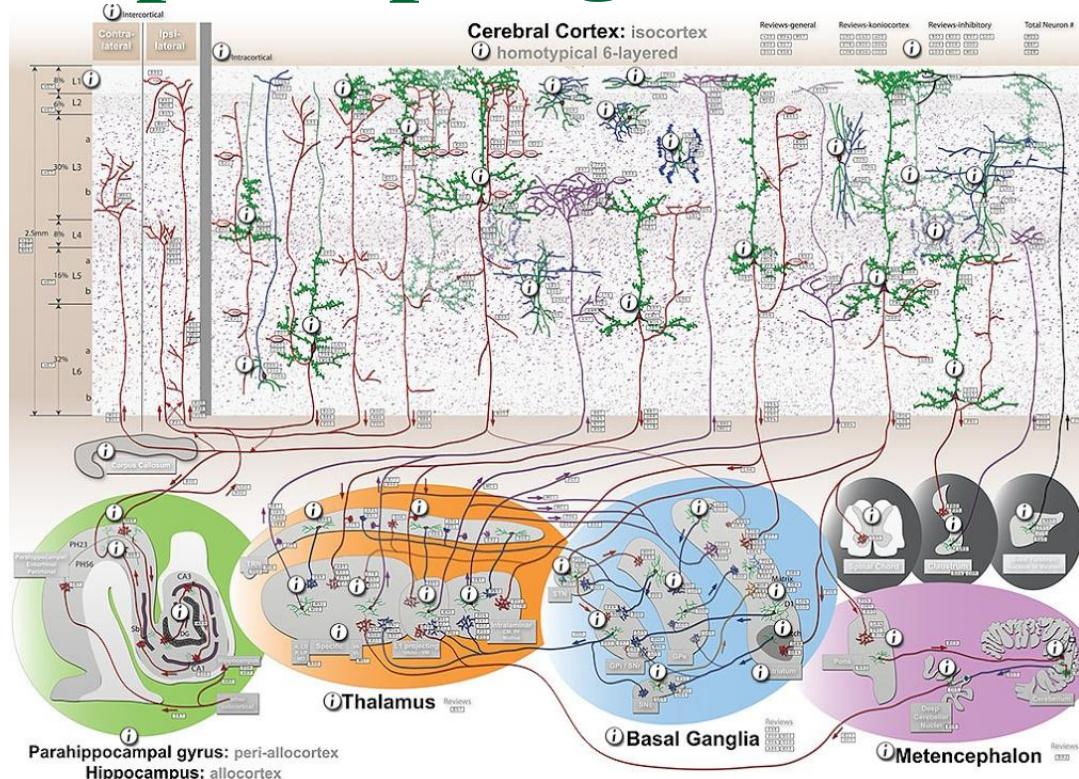
Изображение из: Goertzel et al. A world survey of artificial brain projects, Part II: Biologically inspired cognitive architectures. Neurocomputing 74(1-3):30-49. 2010.

Когнитивные Архитектуры



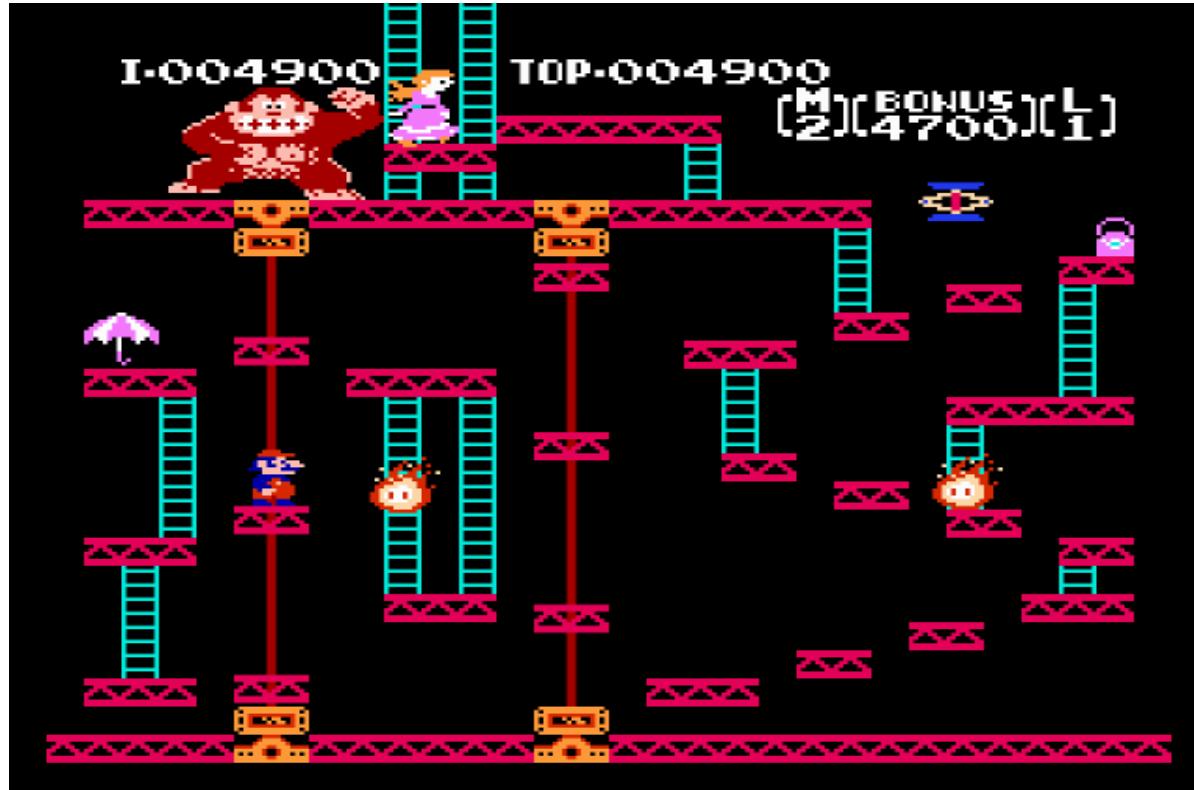
- Модельный/безмодельный подходы
- Проблема привязки символов
- Символьный/субсимвольный разрыв

Просто пример: agi.io



<https://agi.io/2015/12/22/how-to-build-a-general-intelligence-circuits-and-pathways/>

Марсиане и Микропроцессоры



Что мы хотим создать?



Фотосинтез — процесс преобразования энергии света в энергию химических связей органических веществ на свету фотоавтотрофами при участии фотосинтетических пигментов



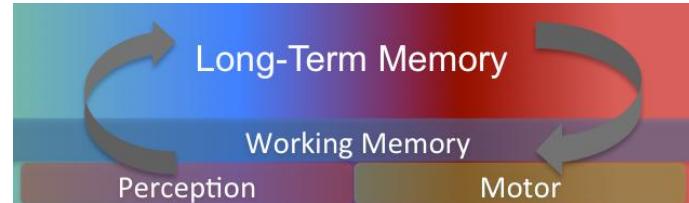
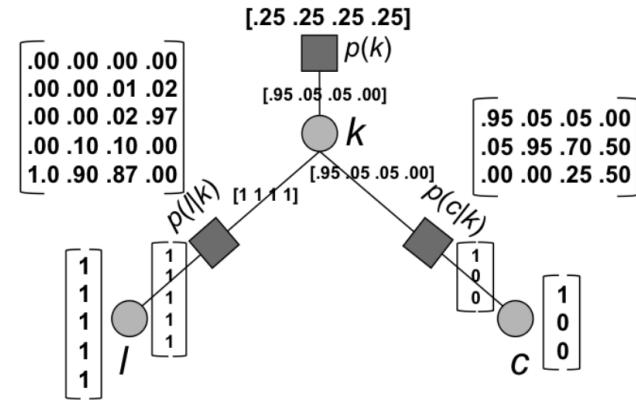
Фотосинтез?

Другой пример: Sigma

- Гипотеза графической архитектуры

- Four desiderata:
 - grand unification
 - generic cognition
 - functional elegance
 - sufficient efficiency

- Деконструкция всех когнитивных функций при использовании фактор-графов как единого «когнитивного субстрата»



Другой пример: Sigma

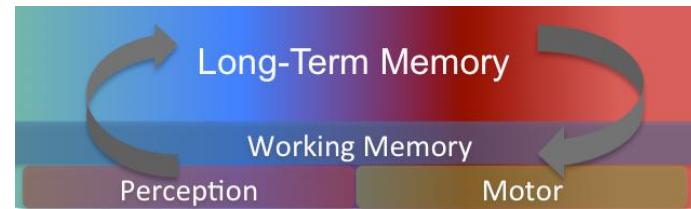
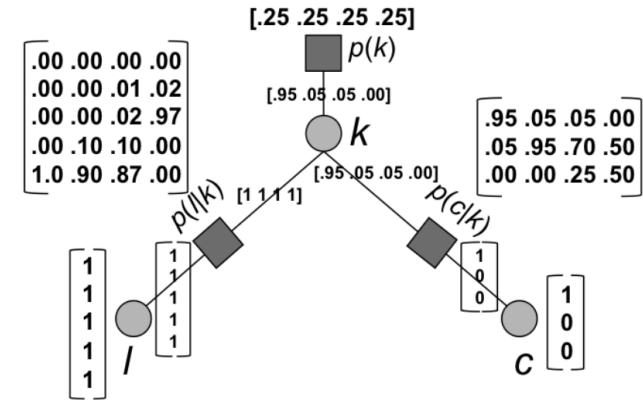
- Гипотеза графической архитектуры

- Four desiderata:

- grand unification
- generic cognition
- functional elegance
- sufficient efficiency

- Деконструкция всех когнитивных функций при использовании фактор-графов как единого «когнитивного субстрата»

Но какие проблемы решаемы?



Что мы хотим создать?

- *Intelligence measures an agent's ability to efficiently achieve goals in a wide range of environments given limited insufficient knowledge and resources*
(B. Goertzel, P. Wang, M. Hutter, etc.)
- Давайте воспримем это определение серьезно

Универсальная индукция

- Универсальные приоры $P(\mu) = 2^{-l(\mu)}$
 μ – программы для универсальной машины Тьюринга
- Условное распределение $P(x|\mu) = \begin{cases} 1, & \text{if } U(\mu) = x^* \\ 0, & \text{otherwise} \end{cases}$
 x – произвольные строки
- МАР $\mu^* = \arg \max_{\mu} P(\mu|x) = \arg \min_{\mu:U(\mu)=x^*} l(\mu|x)$
- Маргинальная вероятность $M_U(x) = \sum_{\mu:U(\mu)=x^*} 2^{-l(\mu)}$
- Предсказание $M_U(y|x) = M_U(xy)/M_U(x)$
- Теорема о бесплатных обедах?

Сходимость!

$$E_Q \left[\sum_{i=1}^n (Q(x_{i+1} = 1 | x_{1:i}) - P_U(x_{i+1} = 1 | x_{1:i}))^2 \right] \leq \frac{\ln 2}{2} K_U(Q)$$

Универсальность пространства алгоритмов

3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164 0628620899
8628034825 3421170679 8214808651 3282306647 0938446095 5058223172 5359408128 4811174502
8410270193 8521105559 6446229489 5493038196 4428810975 6659334461 2847564823 3786783165
2712019091 4564856692 3460348610 4543266482 1339360726 0249141273 7245870066 0631558817
4881520920 9628292540 9171536436 7892590360 0113305305 4882046652 1384146951 9415116094
3305727036 5759591953 0921861173 8193261179 3105118548 0744623799 6274956735 1885752724
8912279381 8301194912 9833673362 4406566430 8602139494 6395224737 1907021798 6094370277
0539217176 2931767523 8467481846 7669405132 0005681271 4526356082 7785771342 7577896091
7363717872 1468440901 2249534301 4654958537 1050792279 6892589235 4201995611 2129021960
8640344181 5981362977 4771309960 5187072113 4999999

```
int a=10000,b,c=8400,d,e,f[8401],g;
main() {for(;b-c;)f[b++]=a/5;
for(;d=0,g=c*2;c-=14, printf("%.4d",e+d/a),e=d%a)
for(b=c;d+=f[b]*a,f[b]=d%--g,d/=g--,--b;d*=b);}

```

By D.T. Winter

Универсальная индукция + RL

- Universal Intelligence Quotient

$$Y(\pi) = \sum_{\mu} 2^{-K(\mu)} V_{\mu}^{\pi} \quad V_{\mu}^{\pi} = E \left[\sum_{t=1}^{\infty} R_t \right]$$

- AIXI как оптимальный интеллектуальный агент
⇒ Строгая постановка задачи для общего ИИ

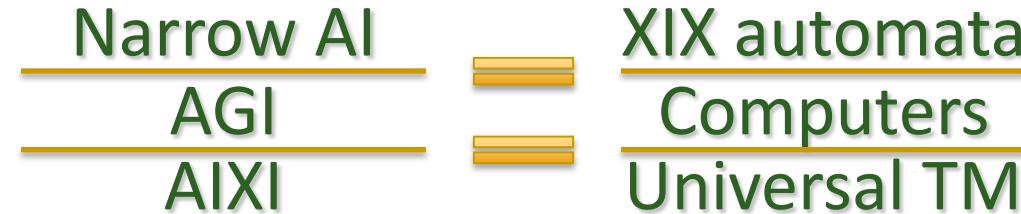
Универсальная индукция + RL

- Universal Intelligence Quotient

$$Y(\pi) = \sum_{\mu} 2^{-K(\mu)} V_{\mu}^{\pi} \quad V_{\mu}^{\pi} = E \left[\sum_{t=1}^{\infty} R_t \right]$$

- AIXI как оптимальный интеллектуальный агент
⇒ Строгая постановка задачи для общего ИИ
- Сильно критикуется
 - Невозможен на практике
 - Не имеет отношения к естественному интеллекту
 - ...который не настолько универсален

Универсальный Интеллект



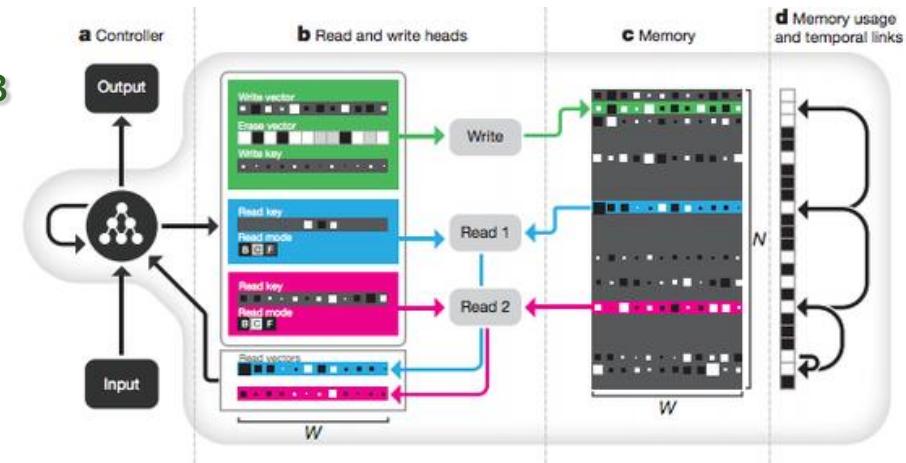
- Если какой-либо подход даже не пытается решить проблему эффективной универсальной индукции математическим или техническим образом, весьма вероятно, что он будет обречен стать очередным узким ИИ

Пример: глубокое обучение

- Слабое обобщение
 - Необходимы большие выборки; нет обучения по одному примеру
 - Невозможность обучения инвариантам
 - Наличие уязвимостей
 - Проблемы с трансферным обучением и обучением без учителя
 - ...
- **Общая причина: неуниверсальное пространство моделей**

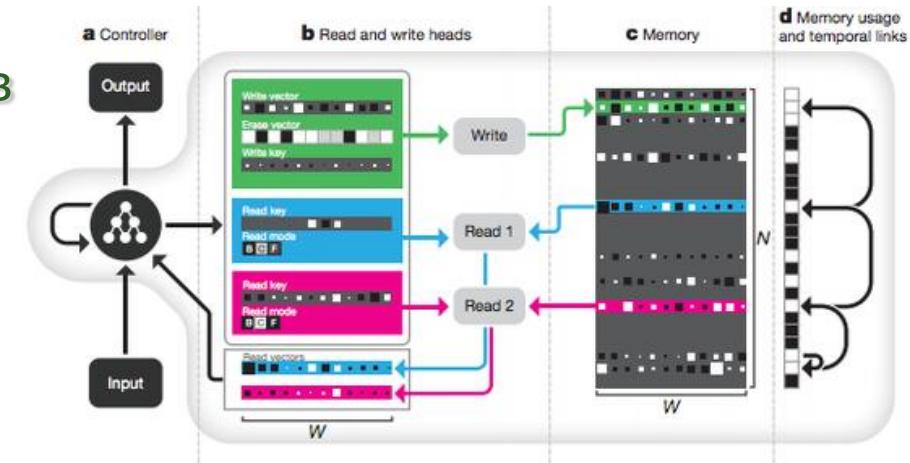
К универсальности в глубоком обучении

- RNN вместо конечных автоматов
- Внешняя память с мягкой адресацией
- Полностью дифференцируемые алгоритмы
- Нейронная машина Тьюринга, нейронная GPU, нейронный программист-интерпретатор, дифференцируемых Forth и т.д.
- + Memory augmented NNs



К универсальности в глубоком обучении

- RNN вместо конечных автоматов
- Внешняя память с мягкой адресацией
- Полностью дифференцируемые алгоритмы
- Нейронная машина Тьюринга, нейронная GPU, нейронный программист-интерпретатор, дифференцируемых Forth и т.д.
+ Memory augmented NNs



=> Как-то работают, но не панацея

Сравнение бэкэндов при обучении алгоритмам

```
const_T = 5

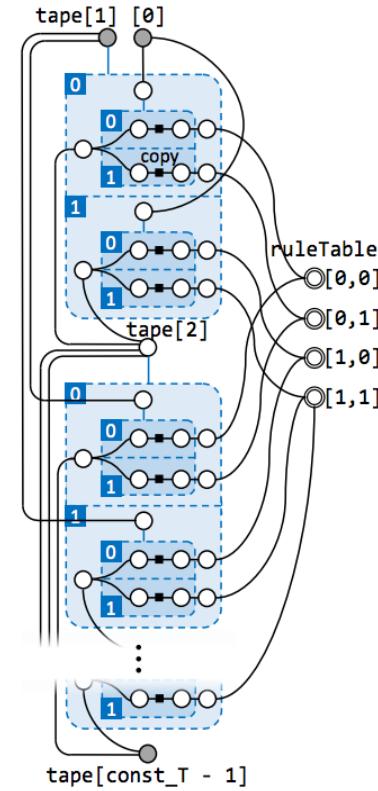
#####
# Source code parametrisation #
#####

ruleTable = Param(2)[2, 2]

#####
# Interpreter model #
#####

tape = Var(2)[const_T]

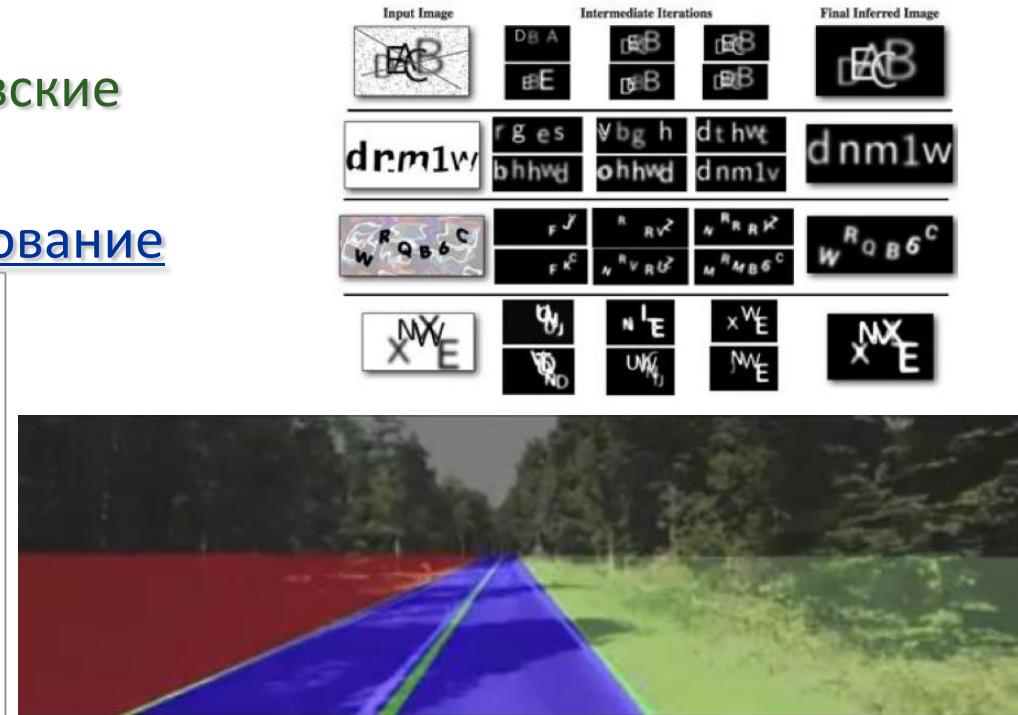
#--IMPORT_OBSERVED_INPUTS--
for t in range(1, const_T - 1):
    with tape[t] as x1:
        with tape[t - 1] as x0:
            tape[t + 1].set_to(ruleTable[x0, x1])
#--IMPORT_OBSERVED_OUTPUTS--
```



Развитие вероятностных моделей

- Графические модели в компьютерном зрении, представлении знаний, глубоком обучении
- Непараметрические байесовские модели
- Вероятностное программирование

```
// Units in arbitrary, uncentered renderer coordinate system.  
ASSUME road_width (uniform_discrete 5 8)  
ASSUME road_height (uniform_discrete 70 150)  
ASSUME lane_pos_x (uniform_continuous -1.0 1.0)  
ASSUME lane_pos_y (uniform_continuous -5.0 0.0)  
ASSUME lane_pos_z (uniform_continuous 1.0 3.5)  
ASSUME lane_size (uniform_continuous 0.10 0.35)  
  
ASSUME eps (gamma 1 1)  
ASSUME port 8000 // External renderer and likelihood server.  
ASSUME load_image (load_remote port "load_image")  
ASSUME render_surfaces (load_remote port "road_renderer")  
ASSUME incorporate_stochastic_likelihood (load_remote port "likelihood")  
ASSUME theta_left (list 0.13 ... 0.03)  
ASSUME theta_right (list 0.03 ... 0.02)  
ASSUME theta_road (list 0.05 ... 0.07)  
ASSUME theta_lane (list 0.01 ... 0.21)  
ASSUME data (load_image "frame201.png")  
ASSUME surfaces (render_surfaces lane_pos_x lange_pos_y lange_pos_z  
    road_width road_height lane_size)  
OBSERVE (incorporate_stochastic_likelihood theta_left theta_right  
    theta_road theta_lane data surfaces eps) True
```



Изображения из: Mansinghka, V., Kulkarni, T., Perov, Y., Tenenbaum, J.: Approximate Bayesian Image Interpretation using Generative Probabilistic Graphics Programs. Advances in NIPS, arXiv:1307.0060 [cs.AI] (2013).

Взлетит ли вероятностное программирование?



SMV/2000=
DL/2012=
PPL/2024???

Функциональное ВП

```
var task = [10, 8, -8, -12, 15, 3]
var target = 1
var generate = function() {
    var subset = repeat(task.length, flip)
    var sum = reduce(function(x, acc)
        { return acc + (x[1] ? x[0] : 0) },
        0, zip(task, subset))
    condition(sum == target)
    return subset
}
Infer({method: "rejection",           → {"probs": [0.48, 0.52],
samples: 100, model: generate})      "support": [[true, true, true, true, false, true],
                                                [true, false, false, true, false, true]]})
```

Рассуждение над знаниями

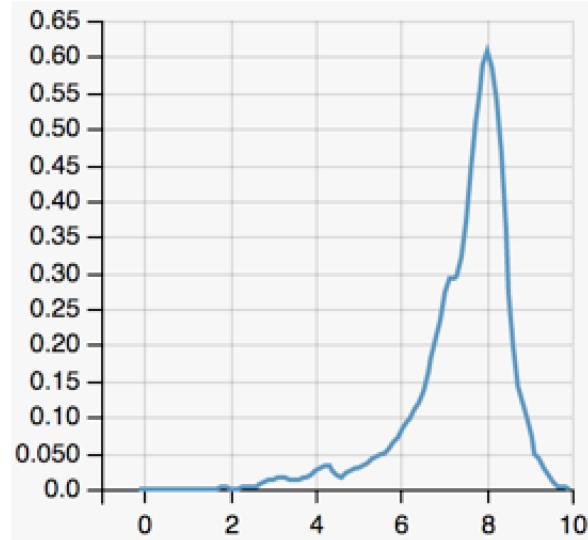
```
var generate = function() {
    var worksInHospital = flip(0.01)
    var smokes = flip(0.2)
    var lungCancer = flip(0.01) || (smokes && flip(0.02))
    var TB = flip(0.005) || (worksInHospital && flip(0.01))
    var cold = flip(0.2) || (worksInHospital && flip(0.25))
    var stomachFlu = flip(0.1)
    var other = flip(0.1)
    var cough = ((cold && flip(0.5)) || (lungCancer && flip(0.3)) || (TB && flip(0.7)) || (other && flip(0.01)))
    var fever = ((cold && flip(0.3)) || (stomachFlu && flip(0.5)) || (TB && flip(0.2)) || (other && flip(0.01)))
    var chestPain = ((lungCancer && flip(0.4)) || (TB && flip(0.5)) || (other && flip(0.01)))
    var shortnessOfBreath = ((lungCancer && flip(0.4)) || (TB && flip(0.5)) || (other && flip(0.01)))
    condition(cough && chestPain && shortnessOfBreath)
    return {lungCancer: lungCancer, TB: TB}
}
```

- Знания как генеративная модель
- Предельное обобщение байесовских сетей

Вероятностное программирование: обучение

- Любыe модели – графические, непараметрические, рекуррентные
- Идентификация моделей, структурное обучение
- Байесовская бритва Оккама бесплатно

```
var xs = [0 , 1 , 2 , 3 ]
var ys = [0.01, 0.99, 4.02, 5.97]
var linreg = function() {
    var a = gaussian(0, 1)
    var b = gaussian(0, 1)
    var sigma = gamma(1, 1)
    var f = function(x) { return a * x + b }
    var check = function(x, y) { observe(Gaussian({mu: f(x), sigma: sigma}), y) }
    map2(check, xs, ys)
    return f(4)
}
Infer({method: 'MCMC', samples: 10000, model: linreg})
```



Вероятностное программирование: нейробайес

```
import edward as ed
from edward.models import Normal
def neural_network(X):
    h = tf.tanh(tf.matmul(X, W_0) + b_0)
    h = tf.tanh(tf.matmul(h, W_1) + b_1)
    h = tf.matmul(h, W_2) + b_2
    return tf.reshape(h, [-1])
W_0 = Normal(loc=tf.zeros([D, 10]), scale=tf.ones([D, 10]))
...
b_2 = Normal(loc=tf.zeros(1), scale=tf.ones(1))
X = tf.placeholder(tf.float32, [N, D])
y = Normal(loc=neural_network(X), scale=0.1 * tf.ones(N))
qW_0 = Normal(loc=tf.Variable(tf.random_normal([D, 10])),
               scale=tf.nn.softplus(tf.Variable(tf.random_normal([D, 10]))))
...
inference = ed.KLqp({W_0: qW_0, ... W_2: qW_2, b_2: qb_2}, data={X: X_train, y: y_train})
```

Вероятностное программирование: нейробайес

```
import edward as ed
from edward.models import Normal
def neural_network(X):
    h = tf.tanh(tf.matmul(X, W_0) + b_0)
    h = tf.tanh(tf.matmul(h, W_1) + b_1)
    h = tf.matmul(h, W_2) + b_2
    return tf.reshape(h, [-1])
W_0 = Normal(loc=tf.zeros([D, 10]), scale=tf.ones([D, 10]))
...
b_2 = Normal(loc=tf.zeros(1), scale=tf.ones(1))
X = tf.placeholder(tf.float32, [N, D])
y = Normal(loc=neural_network(X), scale=0.1 * tf.ones(N))
qW_0 = Normal(loc=tf.Variable(tf.random_normal([D, 10])),
               scale=tf.nn.softplus(tf.Variable(tf.random_normal([D, 10]))))
...
inference = ed.KLqp({W_0: qW_0, ... W_2: qW_2, b_2: qb_2}, data={X: X_train, y: y_train})
```

- Удобно, но немножко не универсально

Обучение вероятностным программам

```
(lambda (stack-id)
  (* 2.0 (* (*
    (* -1.0 (safe-uc 0.0 2.0))
    (safe-uc (safe-uc 4.0
      (+ (safe-log 2.0) -1.0))
    (* (safe-div 2.0
      -55.61617747203855)
    (if (< (safe-uc
      (safe-uc
        27.396810474207317
        (safe-uc -1.0 2.0)) 2.0) 2.0)
      4.0 -1.0)))) -1.0)))
```

```
(lambda (par stack-id) (* (begin (define sym0 0.0)
  (exp (safe-uc -1.0 (safe-sqrt (safe-uc
    (safe-div (safe-uc 0.0 (safe-uc 0.0 3.14159)))
    par) (+ 1.0 (safe-uc (begin (define sym2
      (lambda (var1 var2 stack-id) (dec var2)))
    (sym2 (safe-uc -2.0 (* (safe-uc 0.0 (begin
      (define sym4 (safe-uc sym0 (* (+ (begin
        (define sym5 (lambda (var1 var2 stack-id)
          (safe-div (+ (safe-log (dec 0.0)) -1.0) var1)))
        (sym5 (exp par) 1.0 0)) 1.0) 1.0)))) (if (< (safe-uc
          par sym4) 1.0) sym0 (safe-uc 0.0 -1.0)))) sym0))
      (safe-div sym0 (exp 1.0)) 0)) 0.0)))))) par)))
```

- Модель обучения вероятностным программам можно представить в виде вероятностной программы
- Нетривиальные вероятностные программы удается вывести, но только достаточно простые

Универсальная индукция и ВП

```
var universal_induction = function() {  
    var expr = generate_program()  
    var output = interpret(expr)  
    condition(output == [0, 1, 0, 1, 0, 1, 0, 1])  
    return expr  
}
```

Универсальная индукция и ВП

```
var universal_induction = function() {  
    var expr = generate_program()  
    var output = interpret(expr)  
    condition(output == [0, 1, 0, 1, 0, 1, 0, 1])  
    return expr  
}
```

- Например, комбинаторная логика

```
var generate_program = function() {  
    if(flip(0.4)) return [gen(), gen()]  
    return categorical([1, 1, 1, 1], ['K', 'S', 0, 1])  
}  
var interpret = function(expr) { ... }
```

Универсальная индукция и ВП

```
var universal_induction = function() {  
    var expr = generate_program()  
    var output = interpret(expr)  
    condition(output == [0, 1, 0, 1, 0, 1, 0, 1])  
    return expr  
}
```

- Например, комбинаторная логика

```
var generate_program = function() {  
    if(flip(0.4)) return [gen(), gen()]  
    return categorical([1, 1, 1, 1], ['K', 'S', 0, 1])  
}
```

```
var interpret = function(expr) { ... }
```

➔ Неэффективно, как и любая другая прямая реализация универсальной индукции

Специализация программ

- Пусть $p_L(x,y)$ – некоторая программа от двух аргументов на языке L
- Специализатор $spec_R$ – это такая программа (на языке R), которая получает на вход p_L и x_0 , так что

$$(\forall y) spec_R(p_L, x_0)(y) = p_L(x_0, y)$$

- $spec_R(p_L, x_0)$ – результат глубокой трансформации p_L , которая может быть гораздо эффективнее исходной p_L

Специализация программ

- Пусть $p_L(x,y)$ – некоторая программа от двух аргументов на языке L
- Специализатор $spec_R$ – это такая программа (на языке R), которая получает на вход p_L и x_0 , так что

$$(\forall y) spec_R(p_L, x_0)(y) = p_L(x_0, y)$$

- $spec_R(p_L, x_0)$ – результат глубокой трансформации p_L , которая может быть гораздо эффективнее исходной p_L

Проекции Футамуры-Турчина

$$(\forall x) spec_R(intL, p_L)(x) = intL(p_L, x)$$

Специализация программ

- Пусть $p_L(x,y)$ – некоторая программа от двух аргументов на языке L
- Специализатор $spec_R$ – это такая программа (на языке R), которая получает на вход p_L и x_0 , так что

$$(\forall y) spec_R(p_L, x_0)(y) = p_L(x_0, y)$$

- $spec_R(p_L, x_0)$ – результат глубокой трансформации p_L , которая может быть гораздо эффективнее исходной p_L

Проекции Футамуры-Турчина

$$(\forall x) spec_R(intL, p_L)(x) = intL(p_L, x)$$

$$(\forall p_L, x) spec_R(spec_R, intL)(p_L)(x) = intL(p_L, x)$$

Специализация программ

- Пусть $p_L(x,y)$ – некоторая программа от двух аргументов на языке L
- Специализатор $spec_R$ – это такая программа (на языке R), которая получает на вход p_L и x_0 , так что

$$(\forall y) spec_R(p_L, x_0)(y) = p_L(x_0, y)$$

- $spec_R(p_L, x_0)$ – результат глубокой трансформации p_L , которая может быть гораздо эффективнее исходной p_L

Проекции Футамуры-Турчина

$$(\forall x) spec_R(intL, p_L)(x) = intL(p_L, x)$$

$$(\forall p_L, x) spec_R(spec_R, intL)(p_L)(x) = intL(p_L, x)$$

$$(\forall intL) spec_R(spec_R, spec_R)(intL) = comp_{L \rightarrow R}$$

Еще одна проекция

- Пусть p_L – некоторая генеративная модель (программа на вероятностном языке L)
- Пусть infer_L – это процедура вывода (интерпретатор языка L)

$$(\forall x) \text{spec}_R(\text{infer}_L, p_L)(x) = \text{infer}_L(p_L, x)$$

```
var inference = function(has_dots, is_bright) {  
    var pL = function() {  
        var gene1 = flip()  
        var gene2 = flip()  
        var gene3 = flip()  
        condition(has_dots == (gene1 && gene2) &&  
                 is_bright == (gene1 || gene3))  
        var poisonous = gene1  
        return poisonous  
    }  
    return Infer({model: pL})  
}
```

Еще одна проекция

- Пусть p_L – некоторая генеративная модель (программа на вероятностном языке L)
- Пусть infer_L – это процедура вывода (интерпретатор языка L)

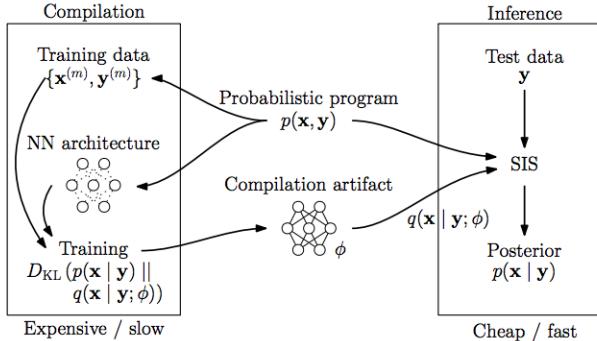
$$(\forall x) \text{spec}_R(\text{infer}_L, p_L)(x) = \text{infer}_L(p_L, x)$$

```
var inference = function(has_dots, is_bright) {  
    var pL = function() {  
        var gene1 = flip()  
        var gene2 = flip()  
        var gene3 = flip()  
        condition(has_dots == (gene1 && gene2) &&  
                 is_bright == (gene1 || gene3))  
        var poisonous = gene1  
        return poisonous  
    }  
    return Infer({model: pL})  
}
```

$\text{spec}_L(\text{Infer}, p_L)$

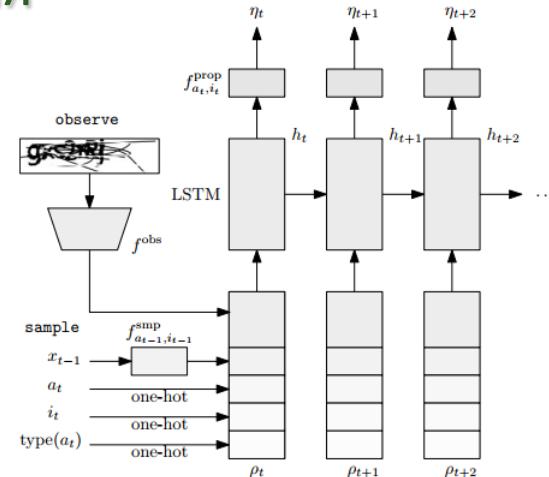
```
var inference =  
function(has_dots, is_bright) {  
    if(!has_dots) {  
        if(!is_bright) return false  
        else return flip()  
    }  
    return true  
}
```

Вариационное приближение или специализация?



- Генеративная модель в форме вероятностной программы «компилируется» в сеть глубокого обучения

```
procedure CAPTCHA
     $\nu \sim p(\nu)$ 
     $\kappa \sim p(\kappa)$ 
    Generate letters:
     $\Lambda \leftarrow \{\}$ 
    for  $i = 1, \dots, \nu$  do
         $\lambda \sim p(\lambda)$ 
         $\Lambda \leftarrow \text{append}(\Lambda, \lambda)$ 
    Render:
     $\gamma \leftarrow \text{render}(\Lambda, \kappa)$ 
     $\pi \sim p(\pi)$ 
     $\gamma \leftarrow \text{noise}(\gamma, \pi)$ 
    return  $\gamma$ 
```



Специализация вывода, или чего не хватает Байесу

- Специализация специализатора по специализатору → бесполезная абстракция
- Специализация специализатора по общей процедуре вывода → искусственный интеллект
- Специализация универсальной индукции по алгоритмически неполной опорной машине → узкий метод машинного обучения
- Специализация общей процедуры вывода по генеративной модели → дискриминантная модель
- Приближенная специализация с одновременным обучением → вариационный вывод

Специализация вывода, или чего не хватает Байесу

- Специализация специализатора по специализатору → бесполезная абстракция
- Специализация специализатора по общей процедуре вывода → искусственный интеллект
- Специализация универсальной индукции по алгоритмически неполной опорной машине → узкий метод машинного обучения
- Специализация общей процедуры вывода по генеративной модели → дискриминантная модель
- Приближенная специализация с одновременным обучением → вариационный вывод
- Частичная специализация → ?

Не только дискриминантные модели

- Дискриминантные модели (прямое вычисление z по x) не всегда возможны, но это не значит, что ситуация безнадежна
- Имеется широкий спектр ‘частичной’ специализации общей процедуры вывода
- Слепой поиск: семплируем решение $z \sim P(z)$; генерируем наблюдения $x \sim P(x|z)$; проверяем совместимость
- Метаэвристический поиск: семплируем $z \sim P(z|z')$ или $z \sim P(z|z', z'')$, где z', z'' – предыдущие кандидаты; P не зависит от задачи
- **Специализированный поиск, управляемый данными:** $z \sim Q_\phi(z|x, z', z'')$, где Q обучается для конкретной задачи
- Дискриминантные модели: $Q_\phi(z|x, z', z'') = Q_\phi(z|x)$

ГП как машина вывода в ВП

1

(flip) = #f
↓
(flip) = #t
↓
(flip) = #f
↓
(flip) = #f
↓
(flip) = #t
↓
↓
'(#t #f)

(define (f)
 (if (flip) '()
 (cons (flip) (f))))))

2

(flip) = #f
↓
(flip) = #t
↓
(flip) = #t
↓
↓
'(#t)

мутация

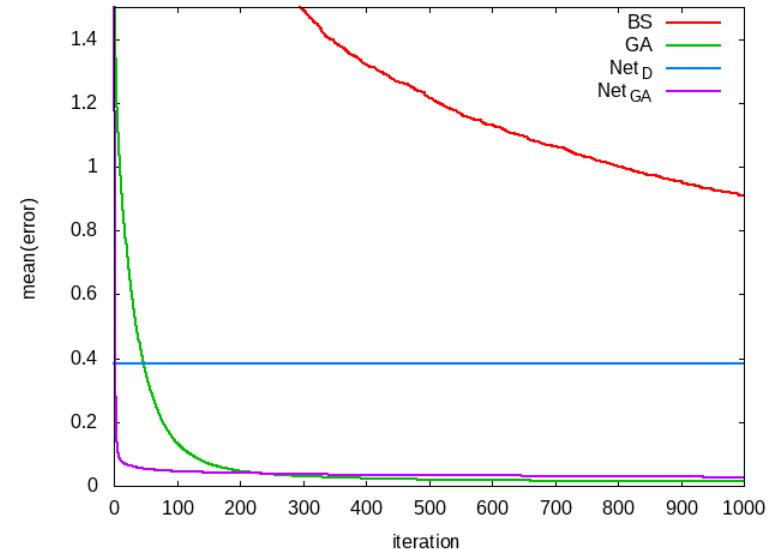
- Генетические операторы
над следами выполнения
вероятностных программ

Пример эксперимента

Задача: $f(\mathbf{z}|\mathbf{A}, \mathbf{b}) = |\mathbf{Az} - \mathbf{b}|^2$

$$\mathbf{z}^* = \mathbf{A}^{-1}\mathbf{b}$$

- Net_D – сеть прямого распространения, которая учится по \mathbf{A} и \mathbf{b} выдавать \mathbf{z}^*
- NetGA – сеть, которая по \mathbf{A} , \mathbf{b} , \mathbf{z}' , \mathbf{z}'' учится выдавать лучший \mathbf{z}
- GA – генетические алгоритмы с обычным скрещиванием
- BS – Brute force search

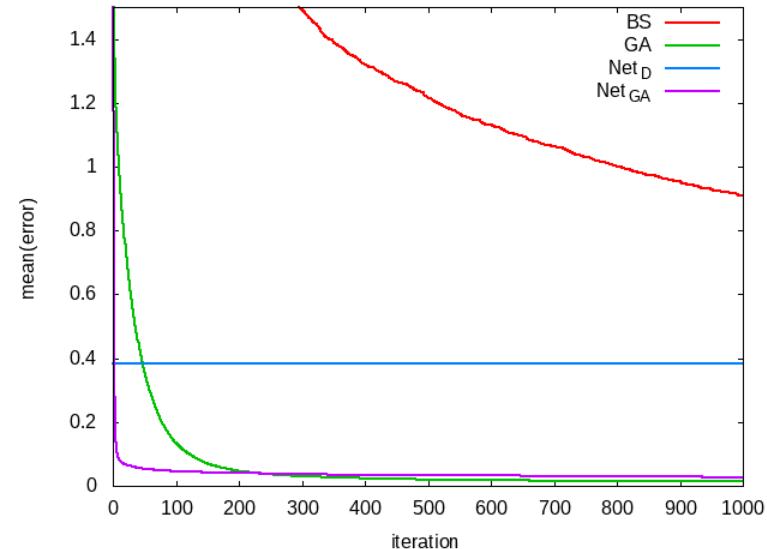


Пример эксперимента

Задача: $f(\mathbf{z}|\mathbf{A}, \mathbf{b}) = |\mathbf{A}\mathbf{z} - \mathbf{b}|^2$

$$\mathbf{z}^* = \mathbf{A}^{-1}\mathbf{b}$$

- Net_D – сеть прямого распространения, которая учится по \mathbf{A} и \mathbf{b} выдавать \mathbf{z}^*
- NetGA – сеть, которая по \mathbf{A} , \mathbf{b} , \mathbf{z}' , \mathbf{z}'' учится выдавать лучший \mathbf{z}
- GA – генетические алгоритмы с обычным скрещиванием
- BS – Brute force search



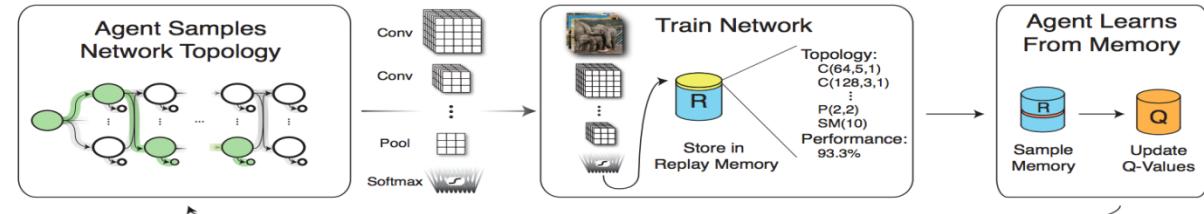
Частичная специализация действительно нужна

Связь с метаобучением

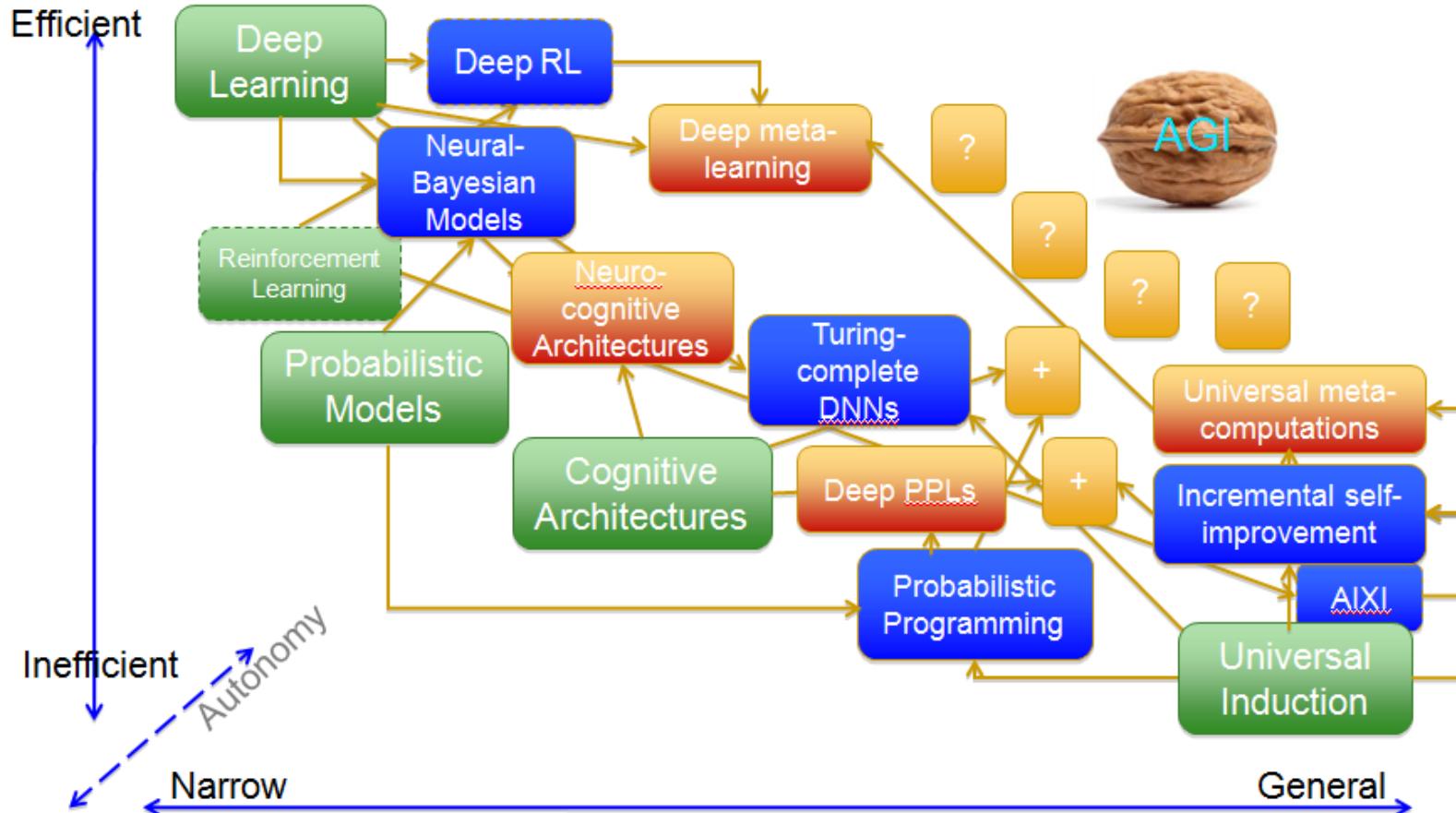
- Рассмотрим задачу обучения нейросети:
 - это задача вывода/поиска
 - градиентный спуск
 - МСМС
 - и т.д.
 - Можно поставить задачу на специализацию этого вывода/поиска
 - Строим мета-сеть, которая учится находить сеть, решающую задачи конкретного класса
 - Полная специализация: прямое отображение параметров задачи в веса сети
 - Частичная специализация: метаобучение

Метаобучение с глубокими сетями

- A neural network that embeds its own meta-levels
- Learning to learn using gradient descent
- Learning to learn by gradient descent by gradient descent
- Learning to reinforcement learn
- RL²: Fast Reinforcement Learning via Slow Reinforcement Learning
- Meta-Learning with Memory-Augmented Neural Networks
- Designing Neural Network Architectures using Reinforcement Learning
- ...



Приближаясь к AGI



Обсуждение

- Глубокое обучение
- Вероятностные модели
- Метавычисления
- Универсальная индукция
- Когнитивные архитектуры
- Инкрементная самооптимизация
- ...
- Общий ИИ



Спасибо за внимание!

alexey@singularitynet.io