Введение в обучение с подкреплением

Тема 10: DQN

Лектор: Кривошеин А.В.

ИНС

Одним из универсальных методов приближения является приближение с помощью **искусственных нейронных сетей** (ИНС). ИНС представляет собой отображение из $\mathbb{R}^d$ в $\mathbb{R}^n$, являющееся композицией линейных и нелинейных отображений. Для более детального описания зафиксируем ряд обозначений. Пусть $h^0 \in \mathbb{R}^d$.

Рассмотрим матрицу W^1 размера $N_1 \times d$ и вектор $b^1 \in \mathbb{R}^{N_1}$. Определим отображение $\mathcal{L}_1$ по правилу:

$z^1 = \mathcal{L}_1(h^0) := W^1 h^0 + b^1 \in \mathbb{R}^{N_1}$.

Далее, к результату применим нелинейную функцию $\varphi_1 : \mathbb{R} \to \mathbb{R}$, называемую функцией активации.

$h^1 = \varphi_1(z^1) \in \mathbb{R}^{N_1}$.

Особенность этой записи в том, что функция активации применяется к вектору поэлементно. Есть множество различных способов выбрать эту функцию (сигмоидальная, ReLU, tanh и т.д.). Композиция двух отображений имеет вид:

$h^1 = \varphi_1(z^1) = \varphi_1(\mathcal{L}_1(h^0)) = [\varphi_1 \circ \mathcal{L}_1](h^0)$.

Эта композиция представляет собой один слой ИНС. Шириной слоя называют число N_1. Эти слои можно наращивать и дальше. Например, для того, чтобы определить i-ый слой рассмотрим матрицу W^i размера $N_i \times N_{i-1}$ и вектор $b^i \in \mathbb{R}^{N_i}$. Тогда

$z^i = \mathcal{L}_i(h^{i-1}) = W^i h^{i-1} + b^i \in \mathbb{R}^{N_i}$ и $h^i = \varphi_i(z^i) = [\varphi_i \circ \mathcal{L}_i](h^{i-1}) \in \mathbb{R}^{N_i}$.

ИНС

ИНС из L слоёв представляет собой композицию:

$$h^L = \Phi(x^0) = (\varphi_L \circ \mathcal{L}_L \ldots \circ \varphi_i \circ \mathcal{L}_i \circ \ldots \circ \varphi_1 \circ \mathcal{L}_1)(h^0),$$

где i-ый слой имеет ширину N_i, $i = 1, \ldots, L$. Удобно такой класс функций обозначить

$$\mathcal{NN}(\#\text{in} = N_0, [N_1, \ldots, N_{L-1}], \#\text{out} = N_L) = \mathcal{NN}(N_0, N_1, \ldots, N_{L-1}, N_L).$$

Здесь $N_0 = d$. Известно, что такого типа функции являются **универсальными аппроксиматорами**. Например, непрерывные функции из $C[0, 1]^d$ можно сколь угодно точно приблизить с помощью ИНС с 2-мя слоями, выбрав достаточную ширину внутреннего слоя. Хорошее приближение достигается за счёт подходящего подбора параметров сети, то есть матриц $W^1, \ldots, W^L$ и векторов $b^1, \ldots, b^L$. Будем обозначать за θ весь набор параметров сети.

ИНС

На практике подбор параметров осуществляется в ходе процесса, называемого **обучением сети**. Для этого формируется обучающий набор данных.

Пусть $g : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ является функцией, которую мы хотим приблизить. Рассмотрим матрицу X размера $M \times N_0$, где M это число обучающих примеров. Каждая строка содержит вектор из $\mathbb{R}^{N_0}$.

За Y обозначим матрицу размера $M \times N_L$ истинных ответов, то есть $Y = \{y_i\}_{i \in M}$, где

$$y_i = g(x_i), \text{ где } x_i \in X.$$

Эти две матрицы X, Y и образуют набор обучающих данных (датасет).

Пусть $\Phi_\theta \in \mathcal{NN}(N_0, N_1, \ldots, N_{L-1}, N_L)$. Цель подобрать параметры θ сети Φ_θ так, что её **ответы на обучающих примерах были как можно ближе к истинным ответам**. Понимать, что значит “близко”, можно по-разному. Например, можно обеспечивать минимум для среднеквадратичной ошибки на данных

$$\text{Loss}(X;\ g, \Phi_\theta) = \frac{1}{2M} \sum_{i=1}^{M} (g(x_i) - \Phi_\theta(x_i))^2.$$

Существует также много иных функций для вычисления ошибки ИНС на данных.

ИНС

Итак, поиск подходящих параметров заключается в минимизации функции ошибки ИНС на данных. Основной численный метод поиска минимума — это **метод градиентного спуска**. Это итеративный метод. Суть в том, чтобы сдвигать параметры сети θ в сторону анти-градиента функции ошибки на малые шаги. Шаг итерации имеет вид:

$$\theta^{k+1} = \theta^k - \alpha \operatorname{grad}_\theta \operatorname{Loss}(X;\, g,\, \Phi_{\theta^k}).$$

где θ_k “старые” параметры, θ^{k+1} новые параметры, α параметр, называемый **шагом обучения** (англ. learning rate). Для среднеквадратичной ошибки на данных шаг итерации имеет вид

$$\theta^{k+1} = \theta^k + \frac{\alpha}{M} \sum_{i=1}^{M} \left(g(x_i) - \Phi\left(x_i,\, \theta^k\right)\right) \operatorname{grad}_\theta \Phi\left(x_i,\, \theta^k\right).$$

Формулы для поиска градиента ИНС по параметрам можно получить, используя формулы дифференцирования сложной функции многих переменных. Этот алгоритм обучения ИНС известен как **алгоритм обратного распространения ошибки**.

Для создания и обучения ИНС будем использовать библиотеку PyTorch.

Суть DQN

Применим ИНС для приближения Q-функции в рамках метода Q-learning. Этот подход называют Deep Q Networks (DQN). Именно метод DQN, применённый в работе 2013 года к обучению агента играть в игры от Atari на уровне превышающем человеческий, стал одним из первых заметных результатов глубокого обучения с подкреплением.

В методе DQN ИНС представляет собой приближение Q-функции, которая по состоянию s выдает приближения Q-функции $\hat{q}(s, a; \theta)$ для возможного дискретного набора действий a.

При оценке фиксированной стратегии π цель обучения ИНС в том, чтобы подобрать параметры θ позволяющие как можно лучше приближать истинные ценности действия $q_\pi(s, a)$:

$\hat{q}(s, a; \theta) \approx q_\pi(s, a)$.

При обучении оптимальной стратегии с помощью метода Q-learning надо подобрать параметры так, чтобы ИНС приближала оптимальные значения $q_*(s, a)$ ценности действий. Обсудим вопрос поиска параметров θ по методу Q-learning.

Суть DQN

Формула обновления Q-функции по методу Q-learning имеет вид

$$Q(S_t, A_t) := Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

Целью обновления является значение $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$.

Используя ИНС для приближения Q-функции мы обновляем не значения Q-функции непосредственно, а параметры θ. Цель обновления в минимизации ошибки между текущим значением Q-функции $\hat{q}(S_t, A_t, \theta)$ и текущим целевым значением $R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \theta)$, например, в следующем виде:

$$J(\theta) = \frac{1}{2}\left(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \theta) - \hat{q}(S_t, A_t, \theta)\right)^2.$$

Полуградиентный шаг обновления параметров имеет вид:

$$\theta_{t+1} = \theta_t + \alpha\left(R_{t+1} + \gamma \max_a \hat{q}(S_t, a, \theta_t) - \hat{q}(S_t, A_t; \theta_t)\right) \mathrm{grad}_\theta\, \hat{q}(S_t, A_t; \theta_t).$$

Однако, процесс обучения по этой формуле не всегда эффективен.

1. Q-learning, как известно, завышает оценки Q-функции относительно истинных.
2. Обучение ИНС по одной паре (S_t, A_t) входных данных не эффективно и не стабильно.
3. Цель обновления не стационарна. ИНС используется для вычисления и прогнозируемого значения и целевого.

Суть DQN

Для решения указанных проблем можно использовать вторую ИНС для вычисления целевого значения. Вторая ИНС имеет ту же структуру, что и исходная ИНС, но с другим набором параметров θ^{-}:

$$J(\theta) = \left(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a; \theta^{-}) - \hat{q}(s, a; \theta)\right)^2.$$

Набор параметров θ^{-} является “замороженной” на некоторое время копией параметров θ. Эти параметры θ^{-} обновляются копированием параметров θ раз в некоторое число итераций. Кроме того, эта модификация превращает полу-градиентный метод формально в полноценный градиентный.

Таким образом, формула обновления по методу Q-learning имеет вид:

$$\theta_{t+1} = \theta_t + \alpha \left(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a; \theta_t^{-}) - \hat{q}(S_t, A_t; \theta_t)\right) \mathrm{grad}_\theta \hat{q}(S_t, A_t; \theta_t).$$

При этом Q-learning позволяет использовать буфер памяти о прошлых взаимодействиях для обучения. Это позволяет проводить обновления параметров не по одному переходу (s, a, r, s'), а использовать пакетное обновление.