Deep Reinforcement Learning for Power Grid Operations

ENERGY 2020 Tutorial

Eric MSP Veith <eric.veith@offis.de>
Motivation
Motivation
Why more AI in the Power Grid?

- Power grid operations increase in complexity
  - More DERs
  - New market concepts, e.g., local markets
  - Ancillary services also from DERs, also market-based

- AI technologies already widespread
  - Forecasting
  - Multi-Agent Systems (mostly rule-based)
  - Distributed heuristics (e.g., schedule planning)

- Resilience: Reaction for the “unknown unknowns”

- Bottom line: **Dynamic strategy development needed**; Deep Reinforcement Learning (DRL) is the next meta-level
A Gentle Introduction to Reinforcement Learning
About Reinforcement Learning

DRL in Relation to other Terms in Deep Learning

- Model-based Learning: ANN develops problem model (vs. Instance-based Learning)
- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
  - Clustering
- Reinforcement Learning
About Reinforcement Learning

DRL in Relation to other Terms in Deep Learning

- **Model-based Learning:** ANN develops problem model (vs. Instance-based Learning)
- **Supervised Learning**
  - Classification
  - Regression
- **Unsupervised Learning**
  - Clustering
- **Reinforcement Learning**

September 20, 2020
About *Reinforcement Learning*

DRL in Relation to other Terms in Deep Learning

- **Model-based Learning**: ANN develops problem model (vs. *Instance-based Learning*)
- **Supervised Learning**
  - Classification
  - Regression
- **Unsupervised Learning**
  - Clustering
- **Reinforcement Learning**

Training Set

September 20, 2020
About Reinforcement Learning

DRL in Relation to other Terms in Deep Learning

- Model-based Learning: ANN develops problem model (vs. Instance-based Learning)
- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
  - Clustering
- Reinforcement Learning
About *Reinforcement Learning*

DRL in Relation to other Terms in Deep Learning

- **Model-based Learning**: ANN develops problem model (vs. *Instance-based Learning*)
- **Supervised Learning**
  - Classification
  - Regression
- **Unsupervised Learning**
  - Clustering
- **Reinforcement Learning**
About Reinforcement Learning

DRL in Relation to other Terms in Deep Learning

> Model-based Learning: ANN develops problem model (vs. Instance-based Learning)

> Supervised Learning
  > Classification
  > Regression

> Unsupervised Learning
  > Clustering

> Reinforcement Learning
About *Reinforcement Learning*

DRL in Relation to other Terms in Deep Learning

- **Model-based Learning:** ANN develops problem model (vs. *Instance-based Learning*)
- **Supervised Learning**
  - Classification
  - Regression
- **Unsupervised Learning**
  - Clustering
- **Reinforcement Learning**
About Reinforcement Learning

DRL in Relation to other Terms in Deep Learning

Artificial Intelligence (AI)
About Reinforcement Learning

DRL in Relation to other Terms in Deep Learning

Machine Learning (ML)

Artificial Intelligence (AI)
About Reinforcement Learning

DRL in Relation to other Terms in Deep Learning

- Deep Learning (DL)
- Machine Learning (ML)
- Artificial Intelligence (AI)
About *Reinforcement Learning*

DRL in Relation to other Terms in Deep Learning

(David Silver)
Basic Terminology
Agent, Sensors, Actuators

> Agent: Acting Entity
> Through Sensors, the Agent perceives its environment
> ...which it acts upon with its Actuators.
Basic Terminology

Agent, Sensors, Actuators: An Example

> **Agent:** Mouse
> **Sensors:** Board (encoding?)
> **Actuators:** Forward, backward, turn $\pm 90^\circ$
**Agent, Sensors, Actuators: An Example**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 2)</td>
<td>(1, 2)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0, 1)</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(1, 0)</td>
<td>(2, 0)</td>
</tr>
</tbody>
</table>

- **Agent:** Vacuum bot
- **Sensoren:** Area immediately in front of the bot
- **Encoding:** 
  
  \[ \text{dirty} \in \{ \text{yes, no} \} \]

- **Actuators:** Forward, backward, turn ±90°
### Basic Terminology

**Agent, Sensors, Actuators: An Example**

<table>
<thead>
<tr>
<th>(0, 2)</th>
<th>(1, 2)</th>
<th>(2, 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 1)</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(1, 0)</td>
<td>(2, 0)</td>
</tr>
</tbody>
</table>

- **Agent:** Vacuum bot
- **Sensoren:** Area immediately in front of the bot
  - **Encoding:**
    - $dirty \in \{\text{yes, no}\}$
  - **Local vs. global**
- **Actuators:** Forward, backward, turn $\pm 90^\circ$
Basic Terminology
Agent, Sensors, Actuators: An Example

> **Agent:** Vacuum bot
> **Sensoren:** Area immediately in front of the bot
> **Encoding:**
>  \[ \text{dirty} \in \{\text{yes, no}\} \]
> **Local vs. global**
> **Sensors noisy?**
> **Actuators:** Forward, backward, turn \( \pm 90^\circ \)
## Basic Terminology

### Agent, Sensors, Actuators: An Example

<table>
<thead>
<tr>
<th>(0, 0)</th>
<th>(1, 0)</th>
<th>(2, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(0, 1)</th>
<th>(1, 1)</th>
<th>(2, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Vacuum Bot" /></td>
<td><img src="image2.png" alt="Area Sensor" /></td>
<td><img src="image3.png" alt="Forward Actuator" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(0, 2)</th>
<th>(1, 2)</th>
<th>(2, 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4.png" alt="Area Sensor" /></td>
<td><img src="image5.png" alt="Encoding" /></td>
<td><img src="image6.png" alt="Slippage" /></td>
</tr>
</tbody>
</table>

> **Agent:** Vacuum bot

> **Sensoren:** Area immediately in front of the bot

> Encoding: $dirty \in \{\text{yes, no}\}$

> Local vs. global

> Sensors noisy?

> **Actuators:** Forward, backward, turn $\pm 90^\circ$

> Slippage?
What route do mouse and bot take?
What route do mouse and bot take?

...or, even more interesting: Why do mouse/bot take a particular route?
Reward
Feedback to the Agent

> **Reward**: Feedback from the environment about the agent’s action regarding the agent’s goal

> “Reward *reinforces* the agent to do the right thing.”

> **Scalar**: Unitless, no further form — big, small, positive, negative, . . .

> No requirements to frequency; most common: per fixed $t$, per action

> **Local**: Rewards the immediate action

> **Training** based on reward (directly or indirectly)

Problem: associating actions and rewards (e.g., bank robbery: high immediate reward, long-term: not so good)
Examples for Reward Values

Stock Trading Profits/Losses

Chess Values of a chess piece, value of a position, result of a game (ELO; or simply win: +1, draw: 0, loss: -1)

Dopamine Level Biological reward: Joy

Vacuum Bot Fill state of the dust tank

Arcade +1 for every frame survived, +1 for every enemy overcome, . . .

Web Crawler Information gain
Examples for Reward Values

Stock Trading  Profits/Losses

Chess  Values of a chess piece, value of a position, result of a game (ELO; or simply win: +1, draw: 0, loss: -1)

Dopamine Level  Biological reward: Joy

Vacuum Bot  Fill state of the dust tank

Arcade  +1 for every frame survived, +1 for every enemy overcome, . . .

Web Crawler  Information gain

Power Grid  Voltage band
Examples for Reward Values

Stock Trading  Profits/Losses

Chess  Values of a chess piece, value of a position, result of a game (ELO; or simply win: +1, draw: 0, loss: -1)

Dopamine Level  Biological reward: Joy

Vacuum Bot  Fill state of the dust tank

Arcade  +1 for every frame survived, +1 for every enemy overcome, . . .

Web Crawler  Information gain

Power Grid  Voltage band, CO₂
Examples for Reward Values

**Stock Trading**  Profits/Losses

**Chess**  Values of a chess piece, value of a position, result of a game (ELO; or simply win: +1, draw: 0, loss: -1)

**Dopamine Level**  Biological reward: Joy

**Vacuum Bot**  Fill state of the dust tank

**Arcade**  +1 for every frame survived, +1 for every enemy overcome, . . .

**Web Crawler**  Information gain

**Power Grid**  Voltage band, CO$_2$, MW from DER
Examples for Reward Values

Stock Trading  Profits/Losses

Chess  Values of a chess piece, value of a position, result of a game (ELO; or simply win: +1, draw: 0, loss: -1)

Dopamine Level  Biological reward: Joy

Vacuum Bot  Fill state of the dust tank

Arcade  +1 for every frame survived, +1 for every enemy overcome, . . .

Web Crawler  Information gain

Power Grid  Voltage band, CO$_2$, MW from DER, line losses avoided
Examples for Reward Values

Stock Trading  Profits/Losses

Chess  Values of a chess piece, value of a position, result of a game (ELO; or simply win: +1, draw: 0, loss: -1)

Dopamine Level  Biological reward: Joy

Vacuum Bot  Fill state of the dust tank

Arcade  +1 for every frame survived, +1 for every enemy overcome, . . .

Web Crawler  Information gain

Power Grid  Voltage band, CO₂, MW from DER, line losses avoided, rel. self-supply, . . .

Caution  Agent maximizes reward — not always the same as succeeding at an objective
Markov Process
Model for Observable Systems

- **System with** $N$ **states**
- **State Space**
  \[ S = \{s_1, s_2, \ldots, s_N\} \]  
- **Markov Property: Chain without memory**
  - Let $Y = (X_t)_{t \in \mathbb{N}}$ be a space of random numbers, $X_t \in S$
  - $Y$ is a markov chain, iff:
    \[ P(X_{t+1} = s_{j_{t+1}} | X_t = s_{j_t}, X_{t-1} = s_{j_{t-1}}, \ldots, X_0 = s_{j_0}) \]
    \[ = P(X_{t+1} = s_{j_{t+1}} | X_t = s_{j_t}). \]
- **Transition Probabilities:**
  \[ p_{ij}(t) := P(X_{t+1} = s_j | X_t = s_i), \quad i,j = 1, \ldots, m \]
- **Transitions Matrix:**
  \[ M(t) = (p_{ij}(t))_{s_i,s_j \in S}, \quad |M| = N \times N \]
> States: *sunny* or *rainy*: \( S = \{s, r\} \)

> History: \([s, s, s, r, s, \ldots]\)

> Probabilities calculated from history: \( M:\)

\[
\begin{array}{cc}
  s & r \\
  s & 0.8 & 0.2 \\
  r & 0.1 & 0.9 \\
\end{array}
\]

Diagram:

- States: sunny (s) and rainy (r)
- Transition probabilities:
  - From sunny to sunny: 0.8
  - From sunny to rainy: 0.2
  - From rainy to sunny: 0.1
  - From rainy to rainy: 0.9

\[s\] (sunny) and \[r\] (rainy) are connected by arrows showing the probabilities of transitions.
use strict;
use warnings;
use Algorithm::MarkovChain;
use Path::Class;
use autodie;  # die if problem reading or writing a file

my @inputs = qw(king_james_bible.txt lovecraft_complete.txt);
my $dir = dir(".");
my $f = ""
my @symbols = ();
foreach $f (@inputs) {
    my $file = $dir->file($f);
    my $lcounter = 0;
    my $wcounter = 0;
    my $file_handle = $file->openr();
    while( my $line = $file_handle->getline() ) {

chomp ($line);
my @words = split( ' ', $line);
push(@symbols, @words);
$lcounter++;
$wcounter += scalar(@words);
}
print "$lcounter lines, $wcounter words read from $f\n";
}
my $chain = Algorithm::MarkovChain::->new();
$chain->seed(symbols => \@symbols, longest => 6);
print "About to spew ...
"
print "---\n"
foreach (1 .. 20) {
    my @newness = $chain->spew(length => 40,
                               complete => [ qw( the ) ]);
    print join (" ", @newness), ".\n\n";
}
Markov Chains III
Fun with texts

$ ./lovebible.pl 2> /dev/null
99820 lines, 821134 words read from king_james_bible.txt
16536 lines, 775603 words read from lovecraft_complete.txt
About to spew ...
---

the backwoods folk -had glimpsed the battered mantel, rickety furniture, and ragged draperies. It spread over it a robber, a shedder of blood, when I listened with mad intentness. At last you know! At last to come to see me. Now Absalom.

(Charlie Stross — http://www.antipope.org/charlie/blog-static/2013/12/lovebiblepl.html)
More Complex Systems
Office Routine

Transition probabilities from observation (count transitions, normalize)

What motivates transitions?

(lapan2018deep)
Markov Reward Process

Where Transitions Come From

- **Transition Probabilities**: System Dynamic
- **Transition Values**: “Belohnung” for a transition
- **Return** of an episode:

\[
G_t = \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{6}
\]

- \(G_t\): Overall Return
- \(R_t\): Reward for a transition at \(t\)
- \(\gamma\): Discount Factor (counts infinite loop)
Discount Factor $\gamma$

How far to look into the Future?

\[ G_t = \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (7) \]

> For each $t$: Calculate return as sum of following rewards $R_t$:

\[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (8) \]

> In eq. (8) $k \to \infty$: Stopping condition?

> Multiplication with $\gamma \in [0.9; 0.99]$: Agent’s “foresight”
Return, Reward, Value
What is a State worth?

- **Reward** from transition
- **Return** at the end of a chain of transitions
- How does an agent choose an action in $s_t$?
Return, Reward, Value

What is a State worth?

- **Reward** from transition
- **Return** at the end of a chain of transitions
- How does an agent choose an action in \( s_t \)?
- **Value**: Expected return for a state

\[
V(s) = \mathbb{E}[G | S_t = s]
\]  
(9)
Return, Reward, Value

What is a State worth?

- **Reward** from transition
- **Return** at the end of a chain of transitions
- How does an agent choose an action in $s_t$?
- **Value**: Expected return for a state

$$V(s) = \mathbb{E}[G|S_t = s]$$  \hspace{1cm} (9)

- For each state $s$,
Return, Reward, Value

What is a State worth?

- **Reward** from transition
- **Return** at the end of a chain of transitions
- How does an agent choose an action in $s_t$?
- **Value**: Expected return for a state

$$V(s) = \mathbb{E}[G|S_t = s]$$ (9)

- For each state $s$,
- is the value of this state, $V(s)$,
What is a State worth?

- **Reward** from transition
- **Return** at the end of a chain of transitions
- How does an agent choose an action in $s_t$?
- **Value**: Expected return for a state

$$V(s) = \mathbb{E}[G|S_t = s] \quad (9)$$

- For each state $s$,
  - is the value of this state, $V(s)$,
  - is the mean (alias expected) return
Return, Reward, Value
What is a State worth?

> **Reward** from transition
> **Return** at the end of a chain of transitions
> How does an agent choose an action in $s_t$?
> **Value**: Expected return for a state

$$V(s) = \mathbb{E}[G|S_t = s]$$ (9)

> For each state $s$,
> is the value of this state, $V(s)$,
> is the mean (alias expected) return
> that follows from the *Markov Reward Process*. 
Return, Reward, Value

An Example: The *Dilbert Reward Process*

> home $\rightarrow$ home : 1 (It’s good to be home.)
> home $\rightarrow$ coffee : 1 (Coffee first!)
> computer $\rightarrow$ computer : 5 (Hard work bears fruit.)
> computer $\rightarrow$ chat : $-3$ (Do not disturb!)
> chat $\rightarrow$ computer : 2 (Back to work.)
> computer $\rightarrow$ coffee : 1 (Coders are catalysts that turn coffee into code.)
> coffee $\rightarrow$ computer : 3 (...)
> coffee $\rightarrow$ coffee : 1 (Good coffee needs time.)
> coffee $\rightarrow$ chat : 2 (Some chat at the coffee maker.)
> chat $\rightarrow$ coffee : 1 (Cup already empty?)
> chat $\rightarrow$ chat : $-1$ (Long conversations become boring fast.)

(lapan2018deep)
Ein Beispiel: Der Dilbert Reward Process

\[
p = 0.3; \; R = 2
\]

\[
p = 0.1; \; R = -3
\]

\[
p = 0.7; \; R = 2
\]

\[
p = 0.2; \; R = 1
\]

\[
p = 0.2; \; R = 3
\]

\[
p = 0.2; \; R = 1
\]

\[
p = 0.2; \; R = 2
\]

\[
p = 0.1
\]

\[
R = 1
\]

\[
p = 0.6
\]

\[
R = 1
\]
Return, Reward, Value

Values of States in the *Dilbert Reward Process*

With $\gamma = 0$:

$> \ V(chat) = -1 \cdot 0.5 + 2 \cdot 0.3 + 1 \cdot 0.2 = 0.3$

$> \ V(coffee) = 2 \cdot 0.7 + 1 \cdot 0.1 + 3 \cdot 0.2 = 2.1$

$> \ V(home) = 1 \cdot 0.6 + 1 \cdot 0.4 = 1.0$

$> \ V(computer) = 5 \cdot 0.5 + (-3) \cdot 0.1 + 2 \cdot 0.2 = 2.6$
Return, Reward, Value

Values of States in the *Dilbert Reward Process*

With $\gamma = 0$:

\[
\begin{align*}
V(chat) &= -1 \cdot 0.5 + 2 \cdot 0.3 + 1 \cdot 0.2 = 0.3 \\
V(coffee) &= 2 \cdot 0.7 + 1 \cdot 0.1 + 3 \cdot 0.2 = 2.1 \\
V(home) &= 1 \cdot 0.6 + 1 \cdot 0.4 = 1.0 \\
V(computer) &= 5 \cdot 0.5 + (-3) \cdot 0.1 + 2 \cdot 0.2 = 2.6
\end{align*}
\]

Most valuable state?
Return, Reward, Value

Values of States in the *Dilbert Reward Process*

With $\gamma = 0$:

- $V(\text{chat}) = -1 \cdot 0.5 + 2 \cdot 0.3 + 1 \cdot 0.2 = 0.3$
- $V(\text{coffee}) = 2 \cdot 0.7 + 1 \cdot 0.1 + 3 \cdot 0.2 = 2.1$
- $V(\text{home}) = 1 \cdot 0.6 + 1 \cdot 0.4 = 1.0$
- $V(\text{computer}) = 5 \cdot 0.5 + (-3) \cdot 0.1 + 2 \cdot 0.2 = 2.6$

Most valuable state? *Computer*:

- $\text{computer} \rightarrow \text{computer}$: common
- $\text{computer} \rightarrow \text{computer}$: high reward
- $\text{computer} \rightarrow \text{computer}$: seldom interrupted

Value for $\gamma = 1$?
Return, Reward, Value
Values of States in the *Dilbert Reward Process*

With $\gamma = 0$:

> $V(\text{chat}) = -1 \cdot 0.5 + 2 \cdot 0.3 + 1 \cdot 0.2 = 0.3$
> $V(\text{coffee}) = 2 \cdot 0.7 + 1 \cdot 0.1 + 3 \cdot 0.2 = 2.1$
> $V(\text{home}) = 1 \cdot 0.6 + 1 \cdot 0.4 = 1.0$
> $V(\text{computer}) = 5 \cdot 0.5 + (-3) \cdot 0.1 + 2 \cdot 0.2 = 2.6$

Most valuable state? *Computer*:

> $\text{computer} \rightarrow \text{computer}$: common
> $\text{computer} \rightarrow \text{computer}$: high reward
> $\text{computer} \rightarrow \text{computer}$: seldom interrupted

Value for $\gamma = 1$? $V(s) = \infty$!

> No *Sink State*
> $V(s) > 0 \ \forall s$
Markov Decision Process

From Observation to Action

- **Markov Process**: States and transition probabilities (Markov Chains)
- **Markov Reward Process**: MP plus value of a state
- ... and now for the decision?!
> **Markov Process:** States and transition probabilities (Markov Chains)

> **Markov Reward Process:** MP plus value of a state

> ... and now for the decision?! Right, that is still missing:

> **Markov Decision Process:** MRP plus Actions

> **Action Space** \( A \) (*action space*): set of actions

\[ A = \{a_1, a_2, \ldots, a_n\} \]
Erweiterung der Transitionsmatrix
Vom Markov Reward Process zum Markov Decision Process

Markov Reward Process

Next State

Current State

$p_{ij}$

Markov Decision Process

Target State

Current State

Action

$p_{ij|k}$
> $p_{ij|k}$ probability for $i \rightarrow j$, if $k$ chosen as action

> $k$ aus Policy:

$$\pi(a|s) = P[A_t = a|S_t = s]$$ (10)

> Formal: Probability distribution over all actions in a given state

> This definition includes random actions during exploration
The Cross-Entropy Method
Based on **Sampling Theorem**
Choosing an Action as Probability Distribution

**Sampling Theorem:**

\[
\mathbb{E}_{x \sim p(x)} \left[ H(x) \right] = \int_x p(x) H(x) \, dx \tag{11}
\]

- \( H(x) \)  Reward from a Policy \( Policy \ x \Leftrightarrow R(\pi(\cdot)) \)
- \( p(x) \)  Distribution over all possible policies

> Maximizing \( H(x) \) by searching all possible distributions (not feasible)
> \( p(x) \) unknown (is the environment)
> Strategy: Iterative development of a distribution \( q(x) \) that approximates \( p(x) \)
Sampling with Distribution

Introducing $q(x)$

Sampling Theorem:

$$\mathbb{E}_{x \sim p(x)} \left[ H(x) \right] = \int x p(x) H(x) \, dx = \int x q(x) \frac{p(x)}{q(x)} H(x) \, dx \quad (12)$$

$$= \mathbb{E}_{x \sim q(x)} \left[ \frac{p(x)}{q(x)} H(x) \right] \quad (13)$$

> In eq. (13) Substituting $p(x) \Leftrightarrow q(x)$
> Goal: Optimization metric (approximation)
> Distance metric between two distributions Kullback Leibler Divergence (KL)
Kullback Leibler Divergence

Distance between \( p(x) \) and \( q(x) \)

\[
KL(p_1(x) \parallel p_2(x)) = \mathbb{E}_{x \sim p_1(x)} \log \frac{p_1(x)}{p_2(x)}
\]

\[
= \mathbb{E}_{x \sim p_1(x)} \left[ \log p_1(x) \right] - \mathbb{E}_{x \sim p_1(x)} \left[ \log p_2(x) \right]
\]

(14)

Alternative Names: Information Gain, relative Entropy

Not symmetric: \( KL(p_1(x) \parallel p_2(x)) \neq KL(p_2(x) \parallel p_1(x)) \), using sums instead:

\[
KL_2(p_1(x) \parallel p_2(x)) = KL_2(p_2(x) \parallel p_1(x)) = KL(p_1(x) \parallel p_2(x)) + KL(p_2(x) \parallel p_1(x))
\]

(15)
Kullback Leibler Divergence
Distance between $p(x)$ and $q(x)$

\[
KL(p_1(x) \parallel p_2(x)) = \mathbb{E}_{x \sim p_1(x)} \left[ \log p_1(x) \right] - \mathbb{E}_{x \sim p_1(x)} \left[ \log p_2(x) \right]
\]

(16)

\[
= \int_{-\infty}^{\infty} p(x) \left( \log p(x) - \log q(x) \right) dx
\]

(17)
Iteratively improving the approximation \( p(x)H(x) \):

\[
q_{i+1}(x) = \arg \min_{q_{i+1}(x)} - \mathbb{E}_{x \sim q_i(x)} \frac{p(x)}{q(x)} H(x) \log q_{i+1}(x)
\]

\[
q_0(x) = p(x) \quad (18)
\]

For Reinforcement Learning:

\[
\pi_{i+1}(a|s) = \arg \min_{\pi_{i+1}} - \mathbb{E}_{z \sim \pi_i(a|s)} \left[ R(z) \geq \psi_i \right] \log \pi_{i+1}(a|s) \quad (19)
\]

\[
> H(x) \iff \left[ R(z) \geq \psi_i \right]
\]

\[
> \text{Indicator Funktion } \left[ R(z) \geq \psi_i \right] = 1 \text{ if reward above threshold, 0 else}
\]

\[
> \text{No normalization — works still}
\]
procedure CrossEntropy(env, batchSize = 16, percentile = 70)
    ann ← GenerateRandomANN()
    for batch ∈ PlayEpisodes(batchSize) do
        obs_e, acts_e, rews_e ← FilterElite(batch, percentile)
        actScores_e ← ann(obs_e)
        loss ← CrossEntropy(actScores_e, acts_e)
        ann ← Optimize(ann, loss)
    end for
end procedure
Influence of Episode Distribution
Pro and Con at the Same Time

Cartpole

Frozen Lake
Influence of Episode Distribution

Pro and Con at the Same Time

Cartpole

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

\[\Sigma = 7\]
\[\Sigma = 4\]
\[\Sigma = 2\]

Frozen Lake

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

\[\Sigma = 1\]
\[\Sigma = 0\]
\[\Sigma = 1\]

70% percentile = 0

September 20, 2020
# Overview CE

## Strengths and Weaknesses of the Cross Entropy Method

### Pros

- **Simplicity:** Easy to understand, implementations in 100 LoC possible
- **Good convergence for short episodes with immediate rewards**

### Cons

- **Episodes must be finite and short**
- **Episodes need high variance in rewards**

### Optimizations:

- **Bigger Batches** (prolonges training)
- **Discount Factor** $\gamma \in [0.9; 0.95]$ favors short episodes (easy to train)
- **Hold Elite Episodes** longer
- **Reduce learning rate during ANN training** (reduces speed of convergence)
The Bellman Principle of Optimality
Value of a State:

\[ V(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R_t \right] \]  

(20)

Example:

> \( V(1) \)? Unknown without \( \pi \)

> Even here infinite states

> Always right:
Value Revisited

Value of a State

Value of a State:

\[ V(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R_t \right] \] (20)

Example:

1 Start \rightarrow 2 End

\[ \text{1,0} \]

2 End \rightarrow 3 End

\[ \text{2,0} \]

3 End

\[ \text{1,0} \]

> \( V(1) \)? Unknown without \( \pi \)

> Even here infinite states

> Always right: \( V(1) = 1.0 \)
Value Revisited

Value of a State

Value of a State:

\[ V(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R_t \right] \]  \( (20) \)

Example:

\begin{align*}
1 \quad \text{Start} & \quad 1,0 \quad \text{End} \\
2 \quad \text{End} & \quad 2,0 \\
3 \quad \text{End} & \quad 2,0
\end{align*}

\[ > \quad V(1)? \text{ Unknown without } \pi \]
\[ > \quad \text{Even here infinite states} \]
\[ > \quad \text{Always right: } V(1) = 1.0 \]
\[ > \quad \text{Always down:} \]
Value Revisited

Value of a State

Value of a State:

\[ V(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R_t \right] \]  \hspace{1cm} (20)

Example:

\[ \begin{array}{c}
1 \text{ Start} \\
\downarrow^{2,0} \\
2 \text{ End} \\
\downarrow^{1,0} \\
3 \text{ End}
\end{array} \]

- \( V(1) \) is unknown without \( \pi \)
- Even here infinite states
  - Always right: \( V(1) = 1.0 \)
  - Always down: \( V(1) = 2.0 \)
Value Revisited

Value of a State

$V(1)$? Unknown without $\pi$

> Even here infinite states
  > Always right: $V(1) = 1.0$
  > Always down: $V(1) = 2.0$
  > $p_{\text{right}} = 0.5$, $p_{\text{down}} = 0.5$:
Value Revisited
Value of a State

1 Start

2 End

3 End

$V(1)$? Unknown without $\pi$

Even here infinite states

Always right: $V(1) = 1.0$
Always down: $V(1) = 2.0$
$p_{right} = 0.5$, $p_{down} = 0.5$:
$V(1) = 1.0 \cdot 0.5 + 2.0 \cdot 0.5 = 1.5$
> $V(1)$? Unknown without $\pi$

> Even here infinite states

> Always right: $V(1) = 1.0$

> Always down: $V(1) = 2.0$

> $p_{\text{right}} = 0.5$, $p_{\text{down}} = 0.5$:

$V(1) =$

$1.0 \cdot 0.5 + 2.0 \cdot 0.5 = 1.5$

> $p_{\text{right}} = 0.1$, $p_{\text{down}} = 0.9$:
> Value Revisited
> Value of a State

1 Start → 2 End

2,0

1,0

3 End

> V(1)? Unknown without $\pi$
> Even here infinite states
  > Always right: $V(1) = 1.0$
  > Always down: $V(1) = 2.0$
  > $p_{\text{right}} = 0.5$, $p_{\text{down}} = 0.5$: $V(1) = 1.0 \cdot 0.5 + 2.0 \cdot 0.5 = 1.5$
  > $p_{\text{right}} = 0.1$, $p_{\text{down}} = 0.9$: $V(1) = 1.0 \cdot 0.1 + 2.0 \cdot 0.9 = 1.9$
Value Revisited
Value of a State

1 Start  1,0  2 End

2,0

3 End

V(1)? Unknown without π

Even here infinite states

Always right: \( V(1) = 1.0 \)
Always down: \( V(1) = 2.0 \)

\( p_{\text{right}} = 0.5, \ p_{\text{down}} = 0.5: \)
\[ V(1) = 1.0 \cdot 0.5 + 2.0 \cdot 0.5 = 1.5 \]

\( p_{\text{right}} = 0.1, \ p_{\text{down}} = 0.9: \)
\[ V(1) = 1.0 \cdot 0.1 + 2.0 \cdot 0.9 = 1.9 \]
Value Revisited

Value of a State

\[ V(1)? \text{ Unknown without } \pi \]

- Even here infinite states
  - Always right: \( V(1) = 1.0 \)
  - Always down: \( V(1) = 2.0 \)
  - \( p_{\text{right}} = 0.5, p_{\text{down}} = 0.5: \)
    \[
    V(1) = 1.0 \cdot 0.5 + 2.0 \cdot 0.5 = 1.5
    \]
  - \( p_{\text{right}} = 0.1, p_{\text{down}} = 0.9: \)
    \[
    V(1) = 1.0 \cdot 0.1 + 2.0 \cdot 0.9 = 1.9
    \]
- And for more than 3 states...?
Value of a State
An abstract Look at $V(s)$

$\begin{align*}
V_0(a = a_k) &= r_k + \gamma V_k \\
V_0 &= \max_{a \in 1...n} (r_a + \gamma V_a)
\end{align*}$
Value of a State
An abstract Look at $V(s)$

$>$ Action 1:

$$V_0(a = a_1) = r_1 + \gamma V_1 \quad (23)$$

$>$ Eine Handlung $i$, stochastisch:

$$V_0(a = a_1) = p_1(r_1 + \gamma V_1) + p_2(r_2 + \gamma V_2) + \cdots + p_n(r_n + \gamma V_n)$$

$$\sum_{i=1}^{n} p_i = 1,0 \quad (24)$$

$>$ Formal für eine beliebige Handlung $a$:

$$V_0(a) = \mathbb{E}_{s \sim S} \left[ r_{s,a} + \gamma V_s \right] = \sum_{s \in S} p_{a,0 \rightarrow s} (r_{s,a} + \gamma V_s) \quad (25)$$
Bellman Principle of Optimality
Finding the Maximum Value of a State

Bellman Equation for deterministic case:

\[ V_0 = \max_{a \in 1...n} \left( r_a + \gamma V_a \right) \]  \hspace{1cm} (26)

Bellman Principle of Optimality:

\[ V_0 = \max_{a \in A} \mathbb{E}_{s \sim S} \left[ r_{s,a} + \gamma V_s \right] = \max_{a \in A} \sum_{s \in S} p_{a,0 \to s} (r_{s,a} + \gamma V_s) \]  \hspace{1cm} (27)
Bellman Principle of Optimality
Finding the Maximum Value of a State

\[ V_0 = \max_{a \in A} \mathbb{E}_{s \sim S} \left[ r_{s,a} + \gamma V_s \right] = \max_{a \in A} \sum_{s \in S} p_{a,0 \rightarrow s} (r_{s,a} + \gamma V_s) \] (28)

> Defining a state’s value as the sum of...
  > *Rewards, r*
  > and *Values* \( V(s) \) of following states \( s \in S \)
  > multiplied by transition probability \( p_{0 \rightarrow s} \)
  > given an action \( a \in A \)

> Applies to *all* \( V(s) \): *Recursion*

> In theory, best action obtainable by complete exploration of the state-action-value space
Recursion, Bellman, & Optimality
Solution to a very real Problem

> Ideal Strategy:

1. Start

2

3

1,0

2,0

Value of a state depends on the following states!

Recursive definition covers all following states (in theory).

(Naive) Policy: For the current state, evaluate all reachable states and choose the action with the biggest value $r + V(s)$. 

September 20, 2020
Ideal Strategy: $1 \rightarrow 3: r = 2$
Ideal Strategy: $1 \rightarrow 3: r = 2$

Or not?! $1 \rightarrow 3 \rightarrow 4: r = -18$

Value of a state depends on the following states!

Recursive definition covers all following states (in theory).

(Naive) Policy: For the current state, evaluate all reachable states and choose the action with the biggest value $r + V(s)$. 
Value of an Action

Value of an action $a$ in State $s$

$$Q_{s,a} = \mathbb{E}_{s'\sim\mathcal{S}} \left[ r_{s,a} + \gamma V_{s'} \right] = \sum_{s'\in\mathcal{S}} p_{a,s\rightarrow s'}(r_{s,a} \gamma V_{s'})$$  \hfill (29)$$

> Expected immediate reward $r_{s,a}$ and discounted long-term reward of the target state

$$V_s = \max_{a\in\mathcal{A}} Q_{s,a}$$  \hfill (30)$$

> Value of a state $s$, $V(s)$, is the value of the best possible action executable in $s$: expressing $V(s)$ via $Q_{s,a}$

$$Q(s, a) = r_{s,a} + \gamma \max_{a'\in\mathcal{A}} Q(s', a')$$  \hfill (31)$$

> Applying the Bellman Principle to actions
Applying the Bellman Principle of Optimality: from Value Iteration to Q Learning
Q Learning
Basis of a Big Family of Algorithms

\[ Q(s, a) = r_{s,a} + \gamma \max_{a' \in A} Q(s', a') \]  \hspace{1cm} (32)

A simple Example:

```
\begin{array}{ccc}
  & s_1 & \\
 s_2 & s_0 & s_4 \\
  & s_3 & \\
\end{array}
```

- \( s_0 \): Initial State
- \( s_1, s_2, s_3, s_4 \): Final States
- \( p = \frac{1}{3} \) per action for slipping left/right
$Q(s, a) = r_{s,a} + \gamma \max_{a' \in A} Q(s', a')$

\begin{align*}
Q(s, a) &= 0 \quad \forall s \in \{1, 2, 3, 4\} \\
Q(s_0, up) &= \frac{1}{3} V_1 + \frac{1}{3} V_2 + \frac{1}{3} V_4 = \frac{1}{3} 1 + \frac{1}{3} 2 + \frac{1}{3} 4 = 2.31 \\
Q(s_0, left) &= \ldots = 1.98 \\
Q(s_0, right) &= \ldots = 2.64 \\
Q(s_0, down) &= \ldots = 2.97 \\
V(s_0) &= \max_{a \in A} Q(s_0, a) = Q(s_0, down) = 2.97
\end{align*}
Q Learning

Q Value the Action Indicator

\[ Q(s, a) = r_{s,a} + \gamma \max_{a' \in A} Q(s', a') \quad (34) \]

> Q better suited than V for selecting actions (value of an action, not value of a state)

> V computable from Q

> Missing: method for calculating Q/V (without knowing all transitions!)

\[
\begin{align*}
Q(s_1, a) &= 0 \\
Q(s_2, a) &= 0 \\
Q(s_3, a) &= 0 \\
Q(s_4, a) &= 0 \\
Q(s_0, up) &= 2.31 \\
Q(s_0, left) &= 1.98 \\
Q(s_0, right) &= 2.64 \\
Q(s_0, down) &= 2.97
\end{align*}
\]
Value Iteration
A naïve Approach to Q Learning

With $\gamma = 0.9$:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$0.9^n$</th>
<th>$\approx$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$0.9^{10}$</td>
<td>0.348</td>
</tr>
<tr>
<td>50</td>
<td>$0.9^{50}$</td>
<td>0.00515</td>
</tr>
<tr>
<td>100</td>
<td>$0.9^{100}$</td>
<td>0.0000265</td>
</tr>
</tbody>
</table>

$r = 2$

$r = 1$

$r = [1, 2, 1, 2, 1, \ldots]$

$V(s_1) = 1 + \gamma(2 + \gamma(1 + \gamma(2 + \ldots)))$

$= \sum_{i=0}^{\infty} 1\gamma2^{2i} + 2\gamma^{2i+1}$

$V(s_2) = 2 + \gamma(1 + \gamma(2 + \gamma(1 + \ldots)))$

$= \sum_{i=0}^{\infty} 2\gamma2^{2i} + 1\gamma^{2i+1}$
Value Iteration
Algorithm in a Nutshell

procedure Valuelteration(env)
    $Q \leftarrow [0]$
    for all $s \in S$, $a \in s$ do
        $Q_{s,a} \leftarrow \sum_{s'} p_{a,s \rightarrow s'} (r_{s,a} + \gamma \max_{a'} Q_{s',a'})$
    end for
    return $Q$
end procedure

State space must be discrete
...and small enough!

Transition probabilities from observations $(s_0, s_1, a)$
Value Iteration
Algorithm in a Nutshell

procedure ValueIteration(env)
    $Q \leftarrow [0]$
    for all $s \in S, a \in s$ do
        $Q_{s,a} \leftarrow \sum_{s'} p_{a,s \rightarrow s'} (r_{s,a} + \gamma \max_{a'} Q_{s',a'})$
    end for
    return $Q$
end procedure

> State space must be discrete
> ... and small enough!
> Transition probabilities from observations ($s_0, s_1, a$)
Deep Q Networks
> Saving \((s, a, r, s')\)
>
> Assumption: every value theoretically known and iterable
>
> Back-of-napkin calculation: 8.5 billion floating point numbers in in 32 GB RAM
> Atari 2600 (Benchmark for DRL): $210 \times 160 = 33600$ pixels, 128 colors

> Each frame: $128^{33600} \approx 10^{70802}$ pictures (states!)

> 99(,9?)% of all iterations nonsensible

> *Space Invaders* & Co not discrete
Motivation
Capacity & Compute Power needed for *Value Iteration*

> Power grid mixed discrete/continuous (tap changer vs. generator scaling)

> State space in quasi-stationary calculations already complex (loaf flow calculations, state estimation, ...)
procedure TabularLearning(env, γ, α)

$Q \leftarrow \emptyset$, $R \leftarrow 0$, $\epsilon_e \leftarrow 1,0$

repeat

$s \leftarrow \text{Read}(env)$

if $s \notin Q \lor \text{random}() < \epsilon_e$ then  \quad \triangleright Exploration vs. Exploitation

$a \leftarrow \text{RandomChoice}(A)$

$\epsilon_e \leftarrow \epsilon_e - 0,02$

else

$a \leftarrow \max_{a \in A} Q_s$

end if

$s', r_s, a \leftarrow \text{Act}(env, a)$

$Q_{s,a} \leftarrow (1 - \alpha)Q_{s,a} + \alpha (r + \gamma \max_{a' \in A} Q_{s',a'})$  \quad \triangleright Bellman

$R' \leftarrow R$

$R \leftarrow R + \gamma r_s, a$

until $|R - R'| < \epsilon_R$

return $Q$

end procedure
Coping with Equivariance
Representing Q as Matrix not Efficient Enough

Equivalent Patterns

> Difference—wrt actions—between both states?
Coping with Equivariance
Representing Q as Matrix not Efficient Enough

> Difference—wrt actions—between both states?
Coping with Equivariance

Representing Q as Matrix not Efficient Enough

Difference—wrt actions—between both states?

None!

But separate entry in $Q_{s,a}$: Regression Problem
Deep Q Learning

Non-Linear Representation for $Q$

Regression Problem: non-linear mapping $f : (s, a) \mapsto Q$

$f$: Artificial Neural Network

Adapting the algorithm:

1. Init $Q(s, a)$ with potentially random approximation
2. $(s, a, r, s') = \text{Act}(env, a)$
3. Calculate error:
   \[
   L = \begin{cases} 
   (Q_{s,a} - r)^2 & \text{at the end of episode,} \\
   (Q_{s,a} - (r + \gamma \max_{a' \in A} Q_{s',a'}))^2 & \text{during the episode.}
   \end{cases}
   \]
   \hspace{1cm} (35)
4. Change $Q(s,a)$ with gradient descent algorithm (Stochastic Gradient Descent, SGD)
5. Repeat from (2) until convergence
Independent and Identically Distributed...?

Base Assumption of SGD a Problem

> Base for Deep Q Learning borrowed from supervised Deep Learning:
> Assumption of SGD: i.i.d
> Neither nor at DRL
  1. Independent: \((s, a, r, s')\) not independent, obviously
  2. Indentically: training data (exploration) differs from optimal policy (exploitation): (exploration vs. exploitation)
> Solution: Replay Buffer
  > Ring buffer
  > fixed size
  > more or less i.i.d., but still “fresh enough”
Correlation between Steps
Achilles’ Heel of the Bellman Principle

\[ Q_{s,a} = r + \gamma \max_{a' \in A} Q_{s',a'} \]  

(36)

> Deriving \( Q_{s,a} \) via \( Q_{s',a'} \): Bootstrapping

> \( s \) and \( s' \) differ in just one step

> Update of \( Q(s, a) \) influences \( Q(s', a') \): Training unstable
  (After updating \( Q(s, a) \), \( Q(s', a') \) becomes worse if immediately explored; next update worsens, etc. ad infinitum)

> **Target Network**: copy of Policy Network for \( Q_{s',a'} \); sync every \( N \) steps

> \( N \) a hyper parameter \( N = [1,000; 10,000] \)
How fast do the invaders move?
> How fast do the invaders move?
>
> *Markov Decision Process* dictates that state is completely derivable from one observation
>
> In RL not always possible:
> Partially Observable Markov Decision Process, POMDP
>
> Hack: Merge $k$ observations (e.g., $k = 4$ frames in ATARI)
procedure DqnLearning(env, $\gamma$, $\alpha$, $N$)

$Q \leftarrow \text{RandomWeights}()$, $\hat{Q} \leftarrow \text{RandomWeights}()$

replayBuffer $\leftarrow []$

$\epsilon \leftarrow 1,0$, $n \leftarrow 0$

repeat

a $\leftarrow \begin{cases} \text{RandomChoice}(A) & \text{if Random()} < \epsilon \\ \arg \max_a Q_{s,a} & \text{else} \end{cases}$

$\epsilon \leftarrow \epsilon - 0.02$

$(s', r) \leftarrow \text{Act(env, a)}$

replayBuffer $\leftarrow$ replayBuffer $\cup (s, a, r, s')$

minibatch $\leftarrow$ RandomSample(replayBuffer)

for all step = $(s, a, r, s') \in$ minibatch do

$y = \begin{cases} r & \text{if EpisodeEnd(minibatch)} \\ r + \gamma \max_{a' \in A} \hat{Q}_{s',a'} & \text{else} \end{cases}$
$\mathcal{L} = (Q_{s,a} - y)^2$

$Q \leftarrow \text{SGD}(Q, y)$

$n \leftarrow n + 1$

if $n = N$ then

$\hat{Q} \leftarrow Q$

$n \leftarrow 0$

end if

end for

until HasConverged()

return $Q$

end procedure
How to Proceed Further
DQN + Extensions (Rainbow Paper) very handy
But suffers from the curse of dimensionality
“Status Quo” for Power Systems: DQN, DDPG
Still a long way in the power systems community until AlphaZero is applied
Power Systems benchmark missing
Framework for multi-agent in power systems missing
Want to help? Drop a note: eric.veith@offis.de