

Command line completion (CLC)

*an illustration of learning and decision making
using the imprecise Dirichlet model (IDM)*

Erik Quaeghebeur & Gert de Cooman

SYSTeMS research group



Classical CLC in action

```
login: erik
```

```
Password:
```

```
Last login: Tue Feb 17 08:24:47 on tty1
```

```
command-prompt$ _
```

Classical CLC in action

```
login: erik
```

```
Password:
```

```
Last login: Tue Feb 17 08:24:47 on tty1
```

```
command-prompt$ log<TAB>
```

logger	login	logname	logout
--------	-------	---------	--------

```
command-prompt$ log_
```

Classical CLC in action

```
login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1
command-prompt$ log<TAB>
logger      login      logname     logout
command-prompt$ logn<TAB>
command-prompt$ logname <ENTER>
erik
command-prompt$ _
```

Classical CLC in action

```
login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1
command-prompt$ log<TAB>
logger      login      logname     logout
command-prompt$ logn<TAB>
command-prompt$ logname <ENTER>
erik
command-prompt$ ls<ENTER>
mail/      logic.dvi   logic.tex
command-prompt$ _
```

Classical CLC in action

```
login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1
command-prompt$ log<TAB>
logger      login      logname     logout
command-prompt$ logn<TAB>
command-prompt$ logname <ENTER>
erik
command-prompt$ ls<ENTER>
mail/      logic.dvi   logic.tex
command-prompt$ dvips log_
```

Classical CLC in action

```
login: erik
Password:
Last login: Tue Feb 17 08:24:47 on tty1
command-prompt$ log<TAB>
logger      login      logname     logout
command-prompt$ logn<TAB>
command-prompt$ logname <ENTER>
erik
command-prompt$ ls<ENTER>
mail/      logic.dvi   logic.tex
command-prompt$ dvips log<TAB>
logic.dvi   logic.tex
command-prompt$ dvips logic.d<TAB>
command-prompt$ dvips logic.dvi _
```

Objective: complementing classical CLC

- Whenever there are multiple completions possible, we want to take the command-history into account.

Objective: complementing classical CLC

- Whenever there are multiple completions possible, we want to take the command-history into account.
- Making the completions user-dependent: build and update a model for the user's behavior.

Objective: complementing classical CLC

- Whenever there are multiple completions possible, we want to take the command-history into account.
- Making the completions user-dependent: build and update a model for the user's behavior.
- To add completion action types, such as
 - returning the 'best guess' completion,
 - listing a set of 'best guesses',
 - listing all possible completions, but ordered.

The user as a multinomial process

Model of the user's behavior:

- A priori, the user has a fixed probability c_{command} of looking for **command**.

The user as a multinomial process

Model of the user's behavior:

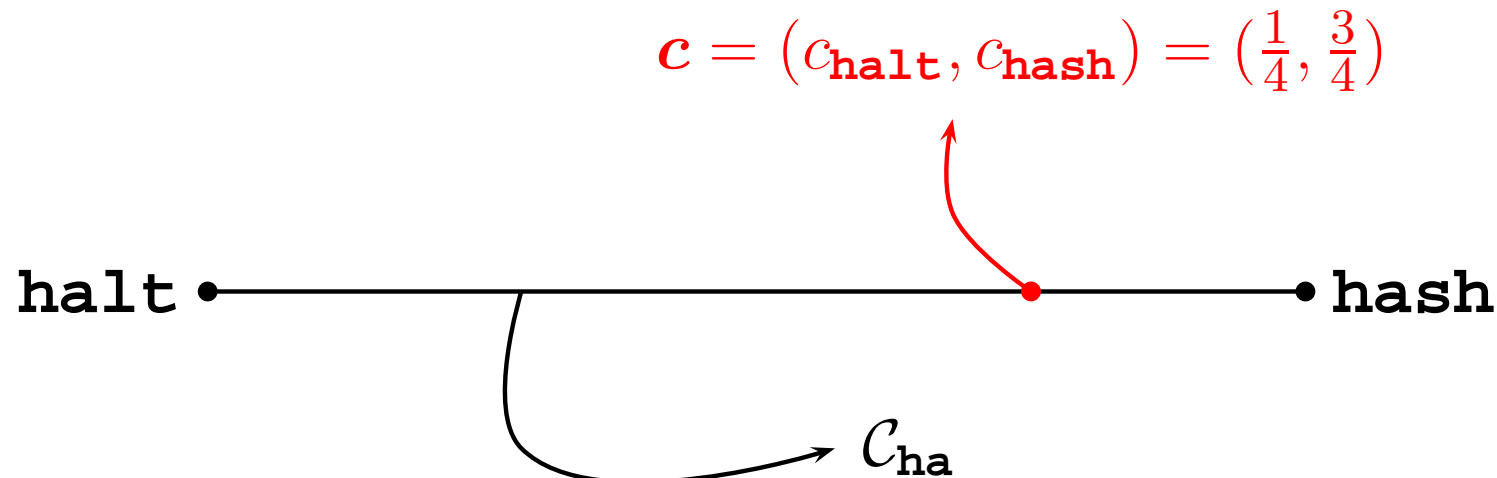
- A priori, the user has a fixed probability c_{command} of looking for **command**.
- After typing part of a command, the remaining possible completions are chosen by the user with the corresponding conditional probabilities.

The user as a multinomial process

Model of the user's behavior:

- A priori, the user has a fixed probability c_{command} of looking for **command**.
- After typing part of a command, the remaining possible completions are chosen by the user with the corresponding conditional probabilities.

Graphical representation of a user:

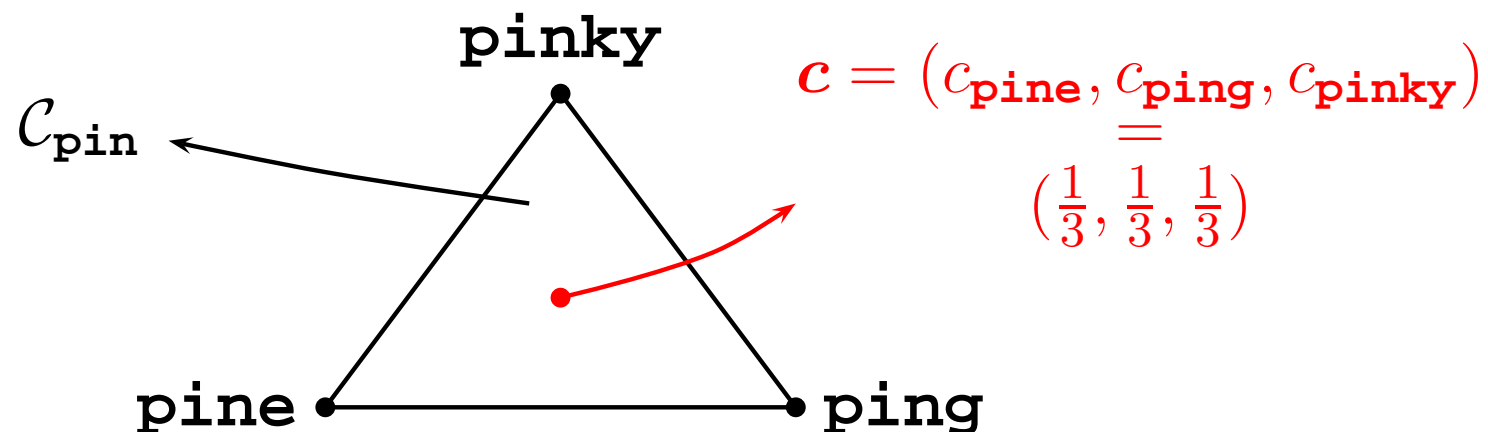


The user as a multinomial process

Model of the user's behavior:

- A priori, the user has a fixed probability c_{command} of looking for **command**.
- After typing part of a command, the remaining possible completions are chosen by the user with the corresponding conditional probabilities.

Graphical representation of a user:



Knowledge about the user's behavior

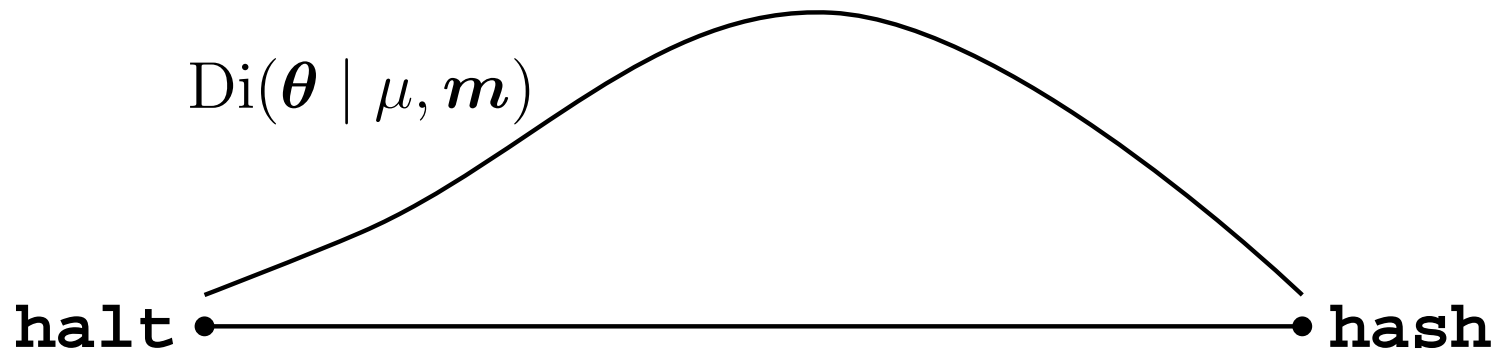
Three models:

1. An exact model: c_{command} is known for all commands.

Knowledge about the user's behavior

Three models:

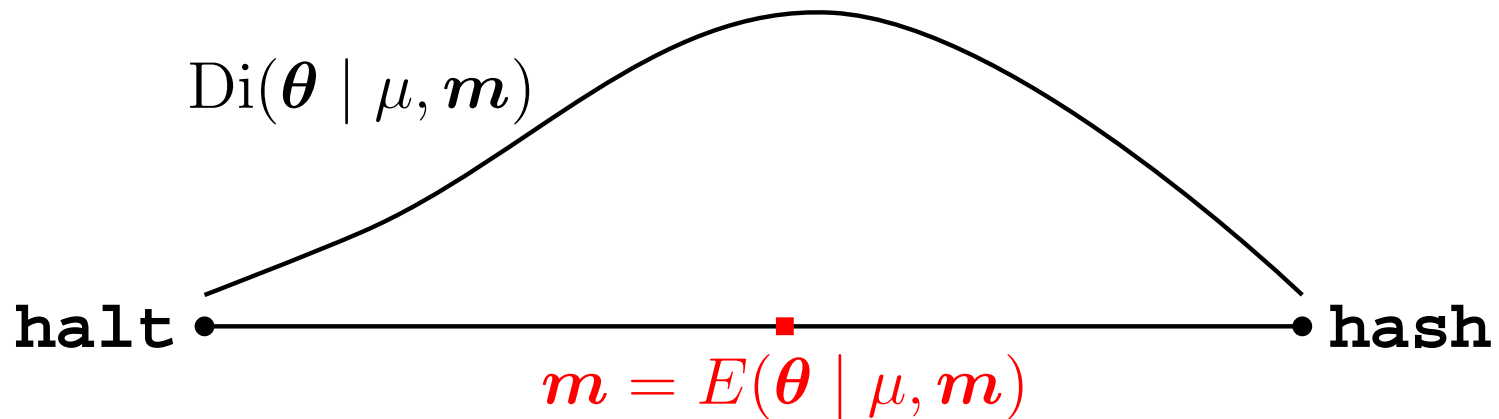
2. A *precise Dirichlet model* (PDM): the uncertainty about the exact model is determined by a Dirichlet distribution.



Knowledge about the user's behavior

Three models:

2. A *precise Dirichlet model* (PDM): the uncertainty about the exact model is determined by a Dirichlet distribution.



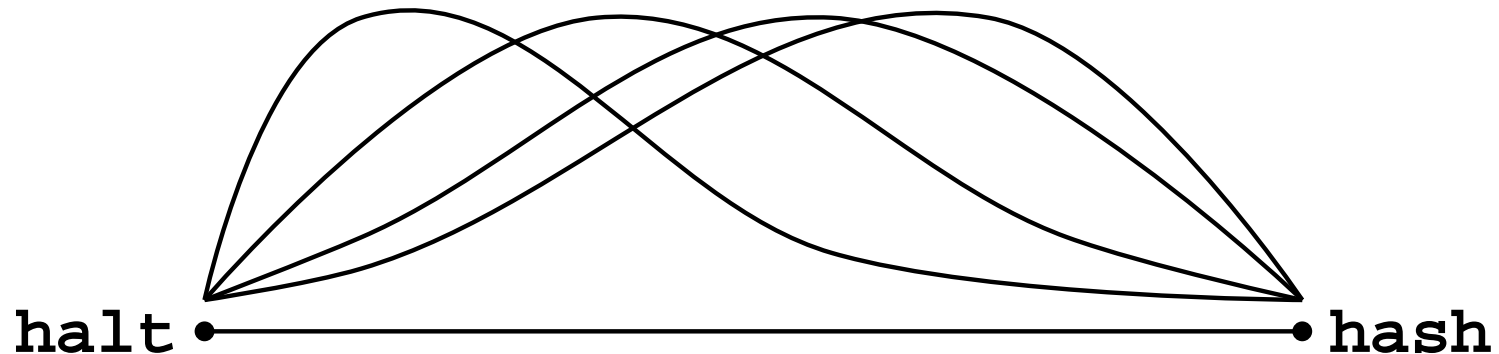
$$\text{Di}(\boldsymbol{\theta} \mid \mu, \mathbf{m}) = \frac{\Gamma(\mu)}{\prod_i \Gamma(\mu m_i)} \prod_i \theta_i^{\mu m_i - 1}$$

$$E(X \mid \mu, \mathbf{m}) = \int_{\mathcal{C}_{\text{ha}}} X(\boldsymbol{\theta}) \text{Di}(\boldsymbol{\theta} \mid \mu, \mathbf{m}) d\boldsymbol{\theta}$$

Knowledge about the user's behavior

Three models:

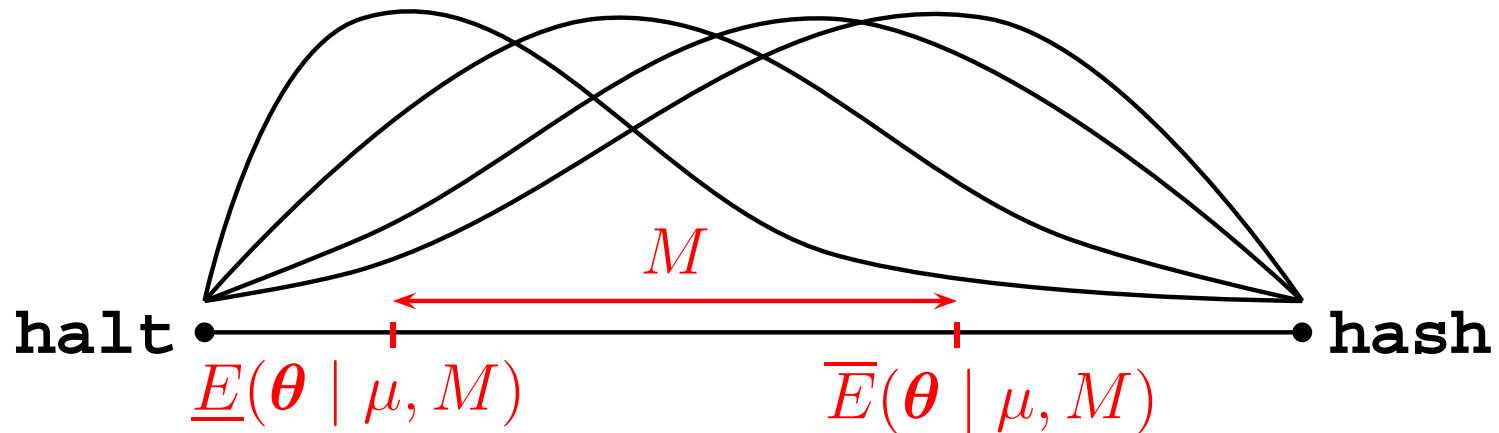
3. An *imprecise Dirichlet model* (IDM): the uncertainty is determined by a set of Dirichlet distributions.



Knowledge about the user's behavior

Three models:

3. An *imprecise Dirichlet model* (IDM): the uncertainty is determined by a set of Dirichlet distributions.



$$\underline{E}(X \mid \mu, M) = \inf_{\mathbf{m} \in M} E(X \mid \mu, \mathbf{m})$$

$$\overline{E}(X \mid \mu, M) = \sup_{\mathbf{m} \in M} E(X \mid \mu, \mathbf{m})$$

From observations to likelihood

- Observations: (a sequence of) executed commands.

From observations to likelihood

- Observations: (a sequence of) executed commands.
- Keep what's relevant for the model, a *sufficient statistic*: the number of occurrences of the commands k , where $\sum_i k_i = \kappa$.

From observations to likelihood

- Observations: (a sequence of) executed commands.
- Keep what's relevant for the model, a *sufficient statistic*: the number of occurrences of the commands k , where $\sum_i k_i = \kappa$.
- *Likelihood function*: likelihood of an exact model given the sufficient statistic, here: a multinomial distribution

$$L_{\mathbf{k}}(\boldsymbol{\theta}) = \frac{\kappa!}{\prod_i k_i!} \prod_i \theta_i^{k_i}.$$

Learning using a PDM/IDM

- Updating a Dirichlet distribution using Bayes' rule:

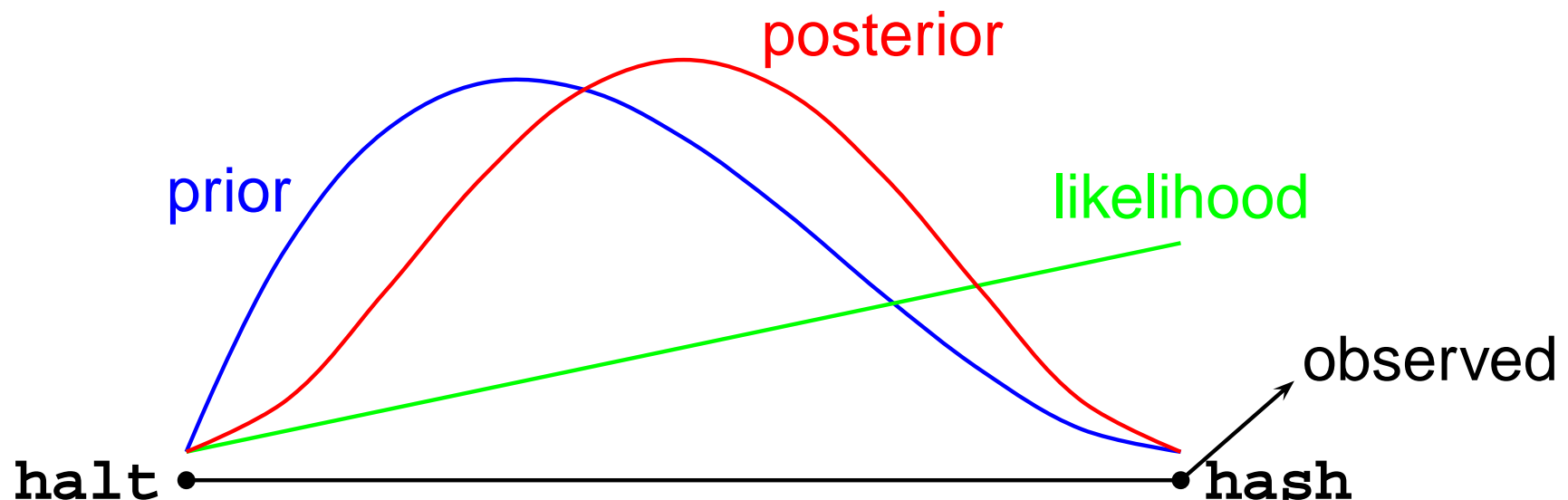
$$\begin{aligned} f(\boldsymbol{\theta} \mid \mu, \mathbf{m}, \mathbf{k}) &\propto \text{Di}(\boldsymbol{\theta} \mid \mu, \mathbf{m}) L_{\mathbf{k}}(\boldsymbol{\theta}) \\ &= \text{Di}(\boldsymbol{\theta} \mid \mu_{\kappa} = \mu + \kappa, \mathbf{m}_{\kappa} = \frac{\mu \mathbf{m} + \mathbf{k}}{\mu + \kappa}). \end{aligned}$$

Learning using a PDM/IDM

- Updating a Dirichlet distribution using Bayes' rule:

$$\begin{aligned} f(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{m}, \boldsymbol{k}) &\propto \text{Di}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{m}) L_{\boldsymbol{k}}(\boldsymbol{\theta}) \\ &= \text{Di}(\boldsymbol{\theta} \mid \boldsymbol{\mu}_{\kappa} = \boldsymbol{\mu} + \boldsymbol{\kappa}, \boldsymbol{m}_{\kappa} = \frac{\boldsymbol{\mu}\boldsymbol{m} + \boldsymbol{k}}{\boldsymbol{\mu} + \boldsymbol{\kappa}}). \end{aligned}$$

Graphically:



Learning using a PDM/IDM

- Updating a PDM is updating the underlying distribution:

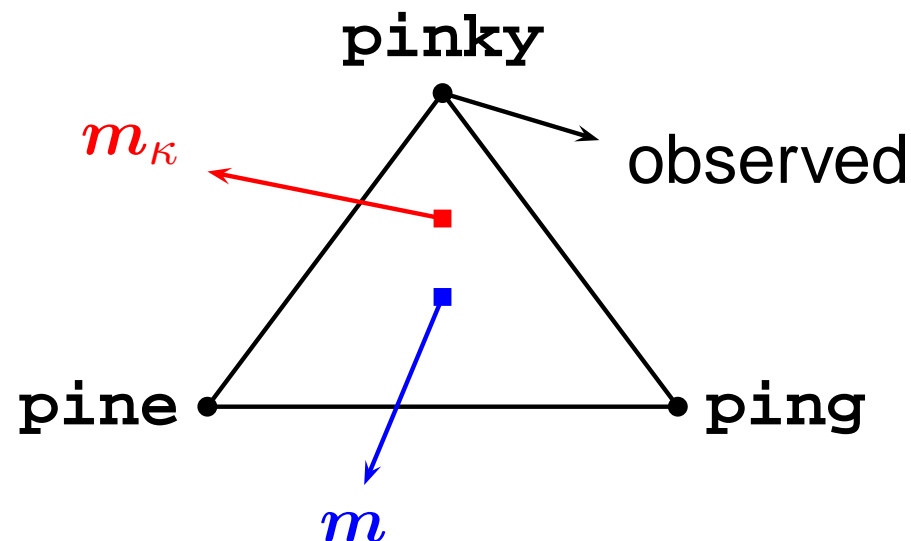
$$E(X \mid \mu, \mathbf{m}) \xrightarrow{k} E(X \mid \mu_k, \mathbf{m}_k).$$

Learning using a PDM/IDM

- Updating a PDM is updating the underlying distribution:

$$E(X \mid \mu, \mathbf{m}) \xrightarrow{k} E(X \mid \mu_{\kappa}, \mathbf{m}_{\kappa}).$$

Graphically:



Learning using a PDM/IDM

- Updating an IDM comes down to updating the corresponding (set of) PDM's:

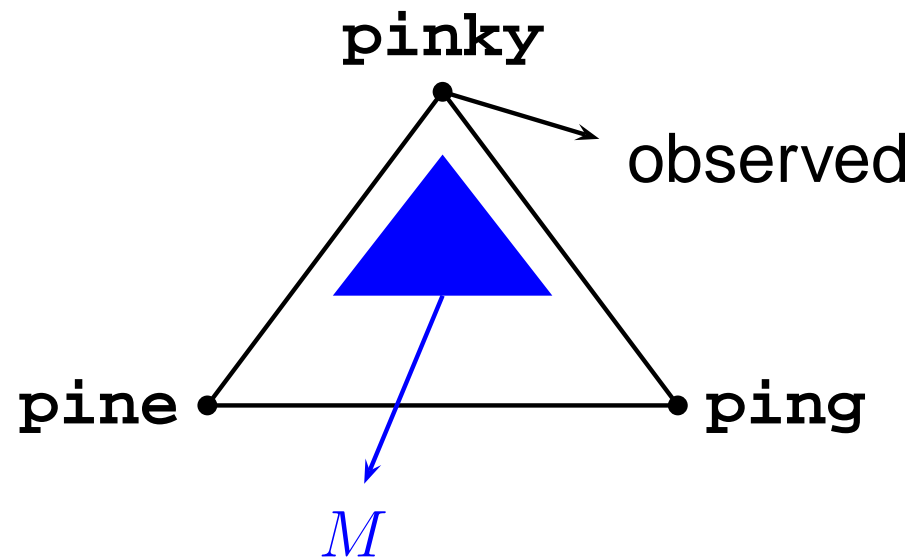
$$\underline{E}(X \mid \mu, M) \xrightarrow{k} \underline{E}(X \mid \mu_{\kappa}, M_{\kappa})$$

Learning using a PDM/IDM

- Updating an IDM comes down to updating the corresponding (set of) PDM's:

$$\underline{E}(X \mid \mu, M) \xrightarrow{k} \underline{E}(X \mid \mu_{\kappa}, M_{\kappa})$$

Graphically:

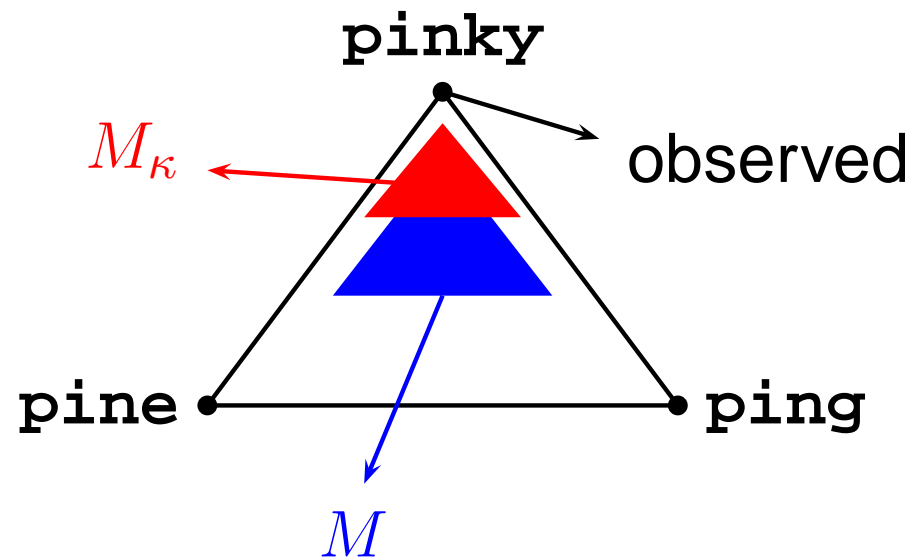


Learning using a PDM/IDM

- Updating an IDM comes down to updating the corresponding (set of) PDM's:

$$\underline{E}(X \mid \mu, M) \xrightarrow{k} \underline{E}(X \mid \mu_{\kappa}, M_{\kappa})$$

Graphically:



Actions and (expected) utility

- Each completion action has a certain utility for the user. Let a gamble X_a correspond to each action a .

Actions and (expected) utility

- Each completion action has a certain utility for the user. Let a gamble X_a correspond to each action a .
- The expected utility of an action:
 - when a PDM is used: $E(X_a \mid \mu, \mathbf{m})$,
 - when an IDM is used: $\underline{E}(X_a \mid \mu, M)$ and $\overline{E}(X_a \mid \mu, M)$.

Actions and (expected) utility

- Each completion action has a certain utility for the user. Let a gamble X_a correspond to each action a .
- The expected utility of an action:
 - when a PDM is used: $E(X_a \mid \mu, \mathbf{m})$,
 - when an IDM is used: $\underline{E}(X_a \mid \mu, M)$ and $\overline{E}(X_a \mid \mu, M)$.
- Choosing one action instead of another also has an expected utility.
This is the expectation of the difference of the corresponding gambles: $E(X_a - X_b)$ or $\underline{E}(X_a - X_b)$.

Decision making: choosing an action

- Ordering the actions based on the expected utility of choosing one action over the other.

Decision making: choosing an action

- Ordering the actions based on the expected utility of choosing one action over the other.
- When using a PDM:
 - compare the actions:
$$a \succ b \iff E(X_a - X_b) > 0 \iff E(X_a) > E(X_b),$$
 - create an ordering of the actions,
 - identify the maximal action(s).

Decision making: choosing an action

- Ordering the actions based on the expected utility of choosing one action over the other.
- PDM: complete ordering of actions.

a — b — $\{c, d\}$ — e — $\{f, g\}$

Decision making: choosing an action

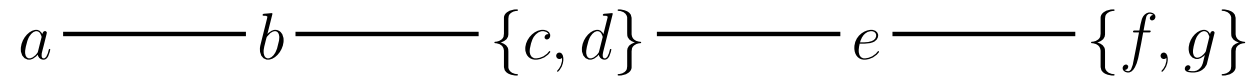
- Ordering the actions based on the expected utility of choosing one action over the other.
- PDM: complete ordering of actions.

$$a \text{ — } b \text{ — } \{c, d\} \text{ — } e \text{ — } \{f, g\}$$

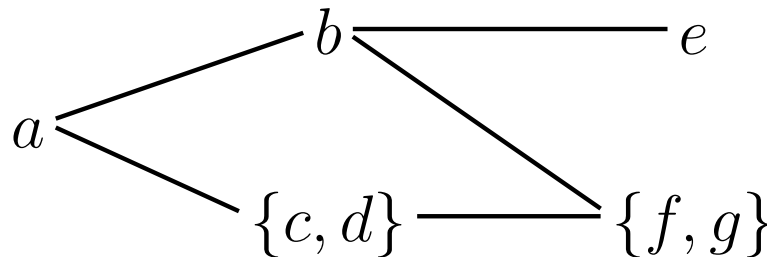
- When using an IDM:
 - compare the actions:
$$a \succ b \iff \underline{E}(X_a - X_b) > 0 \iff \overline{E}(X_b - X_a) < 0,$$
 - create an ordering of the actions,
 - identify the maximal action(s).

Decision making: choosing an action

- Ordering the actions based on the expected utility of choosing one action over the other.
- PDM: complete ordering of actions.



- IDM: partial ordering of actions.



Choosing a unique action

- Choosing the maximal action:
 - Is there a unique maximal action?
 - If there is a unique maximal action: use it.

Choosing a unique action

- Choosing the maximal action:
 - Is there a unique maximal action?
 - If there is a unique maximal action: use it.
- The need for a default action:
 - Whenever there isn't a maximal action, then
 - use the default action.

IDM-CLC in action

```
login: erik
```

```
Password:
```

```
Last login: Wed Feb 18 09:25:48 on tty2
```

```
command-prompt$ _
```


IDM-CLC in action

```
login: erik
```

```
Password:
```

```
Last login: Wed Feb 18 09:25:48 on tty2
```

```
command-prompt$ pin<TAB>
```

```
pine      pinky
```

```
ping
```

```
command-prompt$ pin_
```

IDM-CLC in action

```
login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2
command-prompt$ pin<TAB>
pine      pinky
        ping
command-prompt$ pinky<ENTER>
erik, logged on since Wed Feb 18 12:13:38
command-prompt$ _
```

IDM-CLC in action

```
login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2
command-prompt$ pin<TAB>
pine      pinky
      ping
command-prompt$ pinky<ENTER>
erik, logged on since Wed Feb 18 12:13:38
command-prompt$ pin<TAB>
command-prompt$ pinky _
```

IDM-CLC in action

```
login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2
command-prompt$ pin<TAB>
pine      pinky
        ping
command-prompt$ pinky<ENTER>
erik, logged on since Wed Feb 18 12:13:38
command-prompt$ pin<TAB>
command-prompt$ hash<ENTER>
... [some text] ...
command-prompt$ _
```

IDM-CLC in action

```
login: erik
Password:
Last login: Wed Feb 18 09:25:48 on tty2
command-prompt$ pin<TAB>
pine      pinky
        ping
command-prompt$ pinky<ENTER>
erik, logged on since Wed Feb 18 12:13:38
command-prompt$ pin<TAB>
command-prompt$ hash<ENTER>
... [some text] ...
command-prompt$ pin<TAB>
ping      pine
        pinky
command-prompt$ pin_
```