NPFL099 - Statistical dialogue systems

Dialogue management

Action selection II

Filip Jurčíček

Institute of Formal and Applied Linguistics
Charles University in Prague
Czech Republic

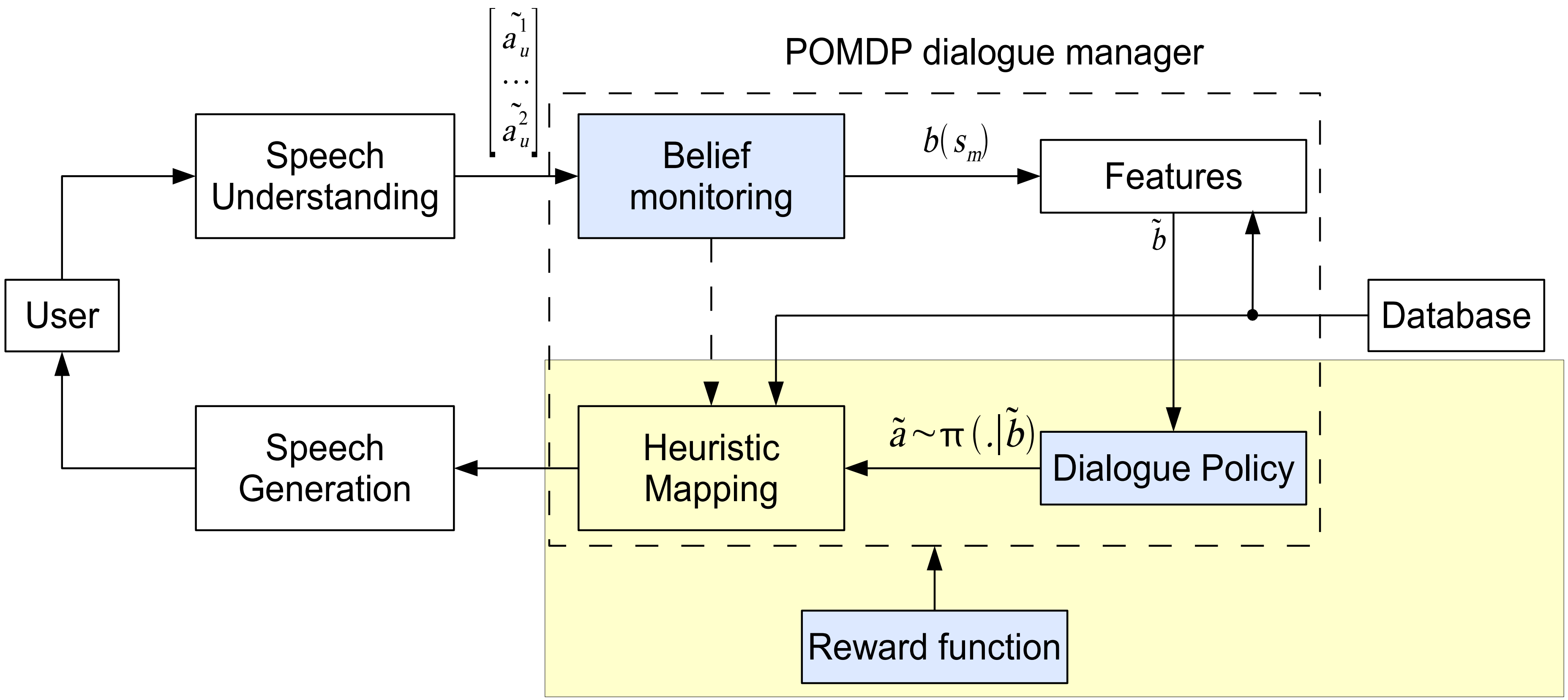
Home page: http://ufal.mff.cuni.cz/~jurcicek

Version: 23/04/2013

Outline

- Stochastic action selection
  - Policy gradients
  - Actor Critics
  - Natural Actor Critic

Action selection



- Ideally the policy is optimized to maximize the reward function

Action selection

- Dialogue policy selects actions given the belief state

- Hand-crafted
  - If $\max_x p(food = x) < 0.3$ then request(food)
  - If $0.3 < \max_x p(food = x) < 0.7$ then confirm(food = x)
  - ...

- Deterministic

$$\pi(b(.;\tau)) = \operatorname*{argmax}_{a'} Q^{\pi}(b(.;\tau), a'; \theta)$$

- Stochastic
  - where θ are the policy parameters

$$a \sim \pi(.|b(.;\tau); \theta)$$

Deterministic policies

- To get a dialogue policy one can define:
  - $Q(s,a)$ - expected future reward of $R = \sum_{t=1, s_1=s, a_1=a} r(s_t, a_t)$
    - for taking action **a**
    - in state **s**

- An optimal policy can be expressed as

$$\pi(s) = \underset{a'}{argmax}\, Q^\pi(s, a')$$

- where

$$Q^\pi(s,a) = \boxed{r(s,a)} + \sum_{s'} \boxed{P(s'|s,a)}\, Q^\pi(s', \pi(s'))$$

This is not generally available!

Therefore it is approximated from data, e.g as in SARSA.

Q-function approximation

- In the previous lecture, we approximated

$$Q^{\pi}(b, a; \theta) \approx \theta^T \cdot \phi(b, a)$$

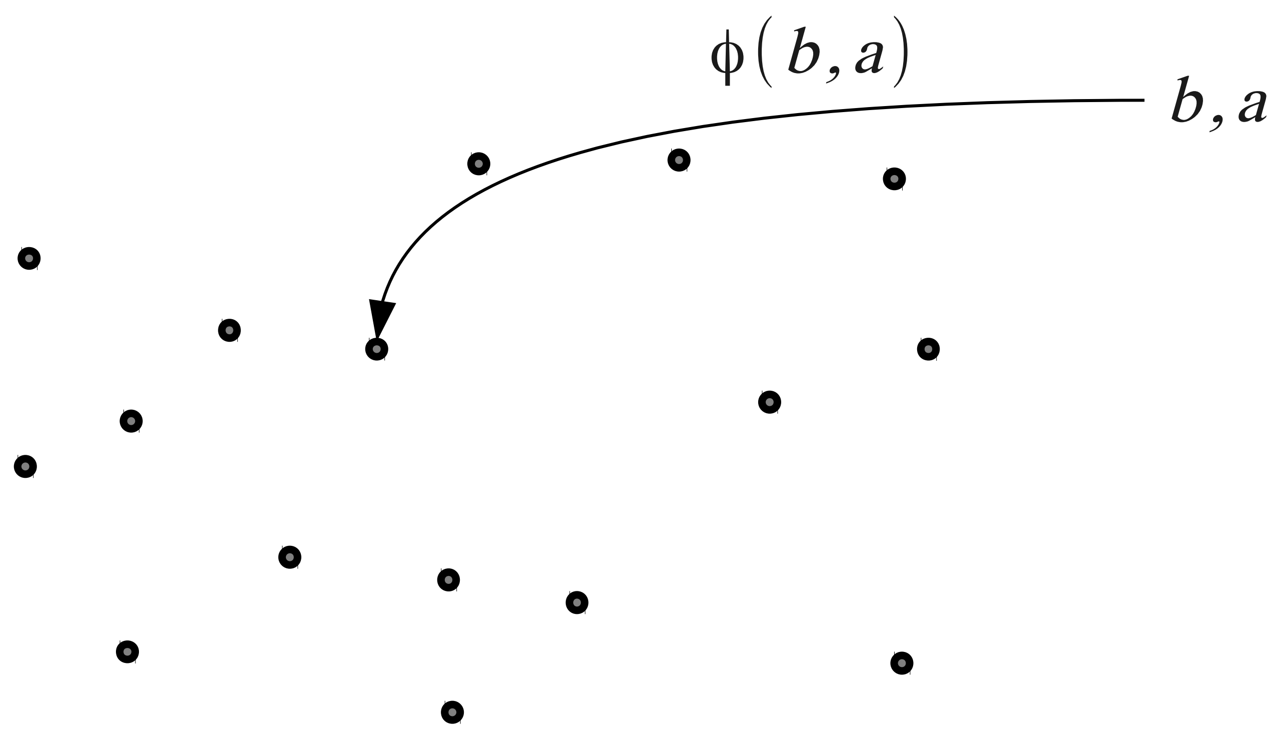
- Then, we used SARSA to compute the approximation

- Instead of parametric methods, non-parametric algorithms can be used

$$Q^{\pi}(b, a; \theta) \approx \theta \cancel{^T \cdot \phi(b, a)}$$

Grid based approach

- In grid based approach
  - Q-function is computed only at discrete points

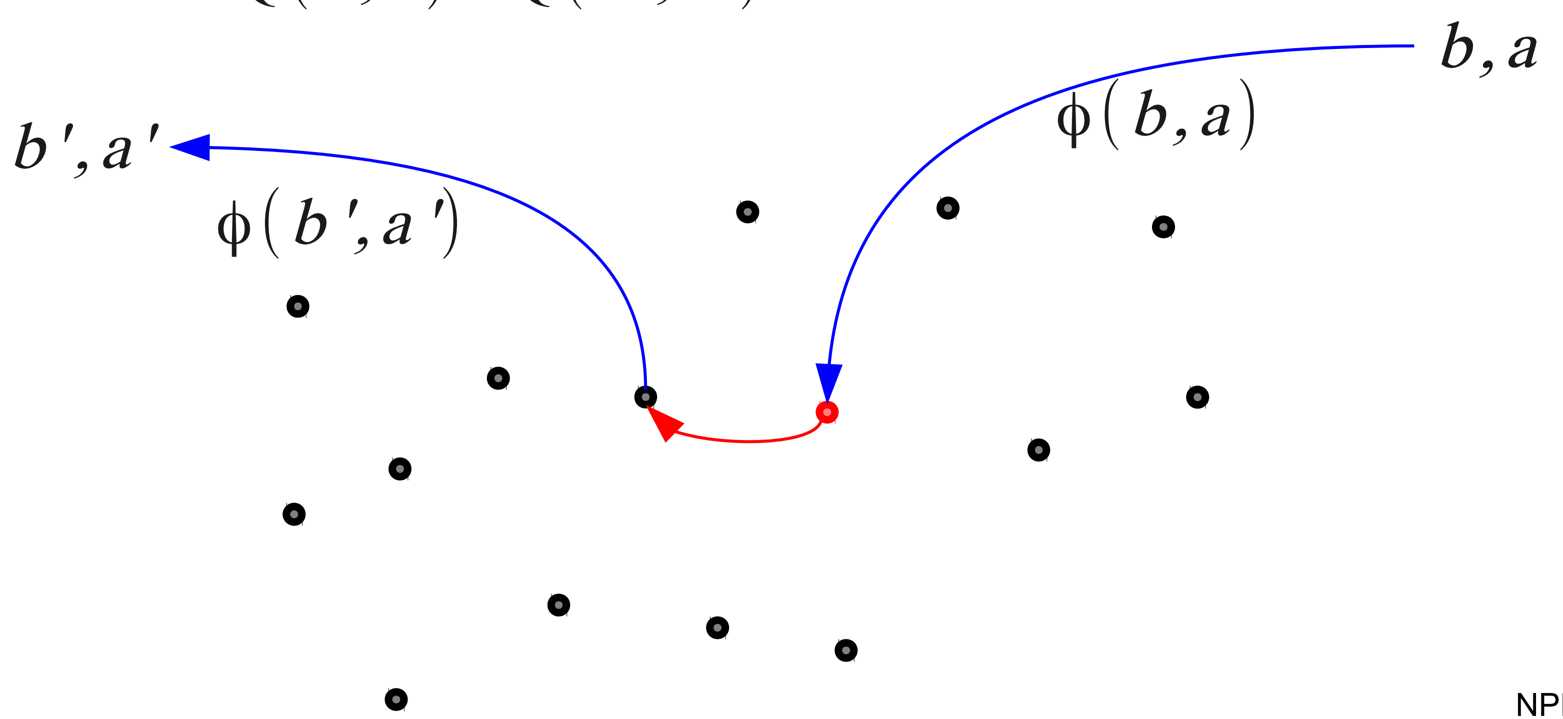


Grid based approach

- Every time we need a Q value for (b,a)

- We find a nearest point $\phi(b',a')$
  - based on some metric $m(\phi(b,a),\phi(b',a'))$
  - and use a Q value at the that nearest point

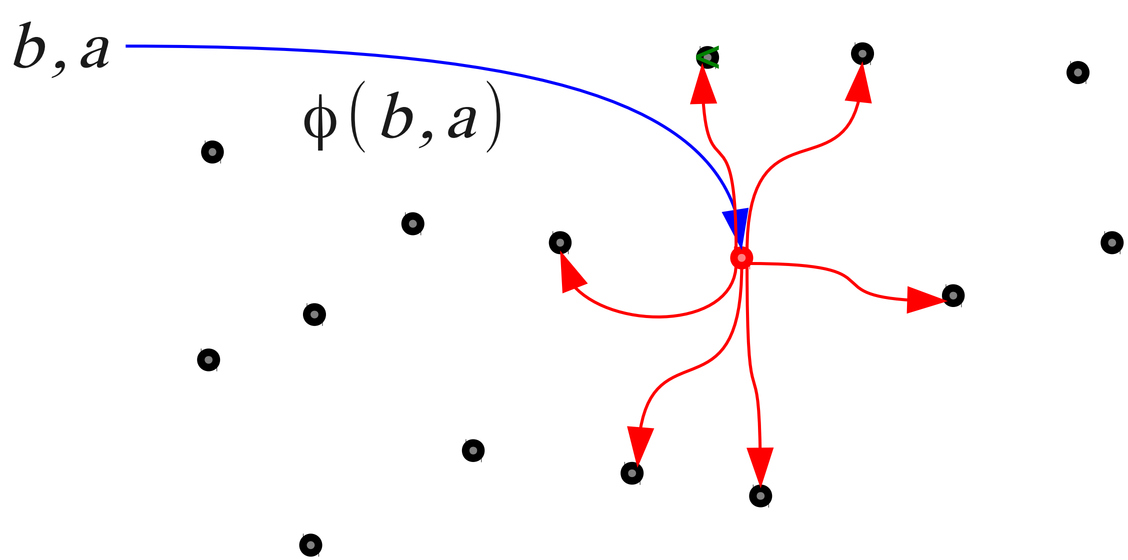
$$Q(b,a) \approx Q(b',a')$$



K-Means approach

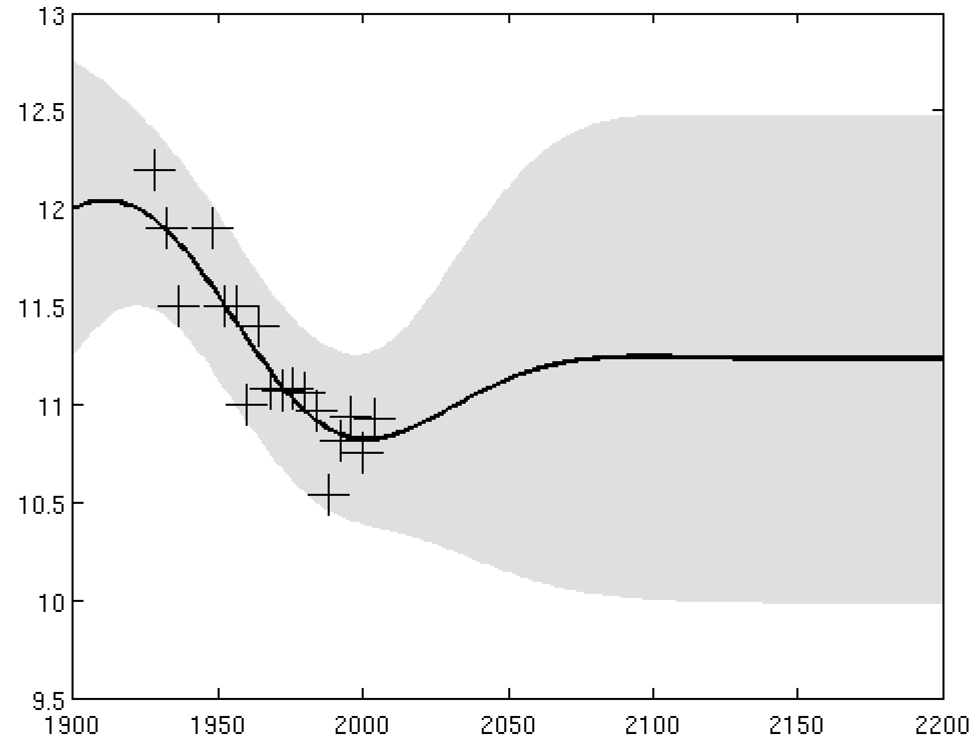
- Although the previous approach looks a bit adhoc
  - it works
- However, more importantly it exhibits some interesting properties
  - it is non-parametric aka K-Means
- In K-Means approach, Q value does not depend only on one (b',a) but on K values which are then averaged

$$Q(b,a) \approx w_1 Q(b',a') + w_2 Q(b'',a'') + \ldots$$



Q function approximation using GP

- This can be generalised using Gaussian Processes
- Gaussian Processes are non-parametric Bayesian approach for regression



- The idea is that Q(b,a) depends on all observed rewards at all visited (b,a) points
  - and we are learning weights of theses grid points

Stochastic policies

- So far, we considered estimation of approximation of a Q-function
  - an expected cumulative reward for taking an action **a** in a belief state **b**
- **This does not have to the most efficient thing to do**
- When modelling a policy
  - we are not really interested in the expected reward
- **We want to know what to do next – an action**

Stochastic policy

- A stochastic policy models directly action selection process

$$a \sim \pi(.|b(.;\tau);\theta)$$

  - where θ are the policy parameters

- Although each turn we sample the actions
  - instead of taking the best action

$$\pi(b(.;\tau)) = \underset{a'}{argmax}\, Q^{\pi}(b(.;\tau), a';\theta)$$

- It can be shown that this can be optimal too

Stochastic policy

- A convenient way to define a stochastic policy is

$$\pi\left(a \middle| b(.;\tau);\theta\right) \propto e^{\theta^T \cdot \Phi\left(a,\, b(.;\tau)\right)}$$

- Again, we use a extracted features to convert POMDP problem to MDP

- The nice property of this Gibbs policy is that it is differentiable with respect to θ

- Remember, we choose the approximation to be **“nice”**

Policy gradients method

- Objective is to maximize expected reward

$$J(\theta, \tau) = E\left[\frac{1}{T}\sum_{t=1}^{T} r(s_t, a_t) \middle| \pi_{\theta,\tau}\right]$$

- Gradient ascent

$$\theta' = \theta + \beta_0 \nabla_\theta J(\theta, \tau)$$

Policy parameters update

$$\tau' = \tau + \beta_1 \nabla_\tau J(\theta, \tau)$$

Model parameters update

Policy gradients method

- Learn to take an action given the observed history

$$\pi(a_t|h_t) \approx \pi(a_t|b(h_t;\tau);\theta)$$

This is what we really want!

- History (observed)

$$h_t = \{a_0, \underbrace{o_1}_{\text{observation}}, a_1, \ldots, \underbrace{a_{t-1}}_{\text{action}}, o_t\}$$

- We observe
  - observations (e.g. users dialogue acts)
  - actions (e.g. systems dialogue acts)

Objective function for POMDPs

- Although we do not observe a state of a dialogue, it is easier if we can work with it

- Full history (trajectory)

$$H_T = \{ h_T, s_{0:T} \}$$

unobserved states

Good for derivation!
All dependencies
on **s** will cancel later

- Equivalent objective function:

$$J(\theta, \tau) = \int p(H; \theta, \tau) R(H) \, dH$$

  - is expected reward of full dialogue history H

Approximation of the gradient

$$\nabla J(\theta,\tau) = \nabla \int p(H;\theta,\tau)\, R(H)\, dH$$

$$= \int \nabla p(H;\theta,\tau)\, R(H)\, dH$$

$$= \int p(H;\theta,\tau)\, \nabla \log p(H;\theta,\tau)\, R(H)\, dH$$

- We used a “log-ratio trick”

$$\nabla \log p(H;\theta,\tau) = \frac{1}{p(H;\theta,\tau)} \nabla p(H;\theta,\tau)$$

Monte Carlo approximation

- Monte Carlo approximation of the gradient

$$\nabla J(\theta,\tau)=\int \boxed{p(H;\theta,\tau)} \nabla \log p(H;\theta,\tau) R(H)\, dH$$

  - by observing dialogues and received rewards

$$\nabla J(\theta,\tau)\approx\frac{1}{N}\sum_{n=1}^{N}\nabla \log p(H^n;\theta,\tau) R(H^n)$$

Probability of a full dialogue history

$$p(H;\theta,\tau) = p(s_0)\prod_{t=1}^{T} p(o_t|s_t)\, p(s_t|a_{t-1}, s_{t-1})\,\pi(a_{t-1}|b(h_{t-1};\tau);\theta)$$

$$\nabla \log p(H;\theta,\tau) = \sum_{t=0}^{T} \nabla \log \pi(a_t|b(h_t;\tau);\theta) + const.$$

- As a result

$$\nabla J(\theta,\tau) \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T_n} \nabla \log \pi(a_t^n|b(h_t^n;\tau);\theta)\, R(H_t^n)$$

Gibbs policy

$$\pi(a|b(.;\tau);\theta) \;\propto\; e^{\theta^T \cdot \Phi(a, b(.;\tau))}$$

- It is easy to get $\nabla_\theta \log \pi(a|b(.;\tau);\theta)$ as it is "linear" in θ
- However, $\nabla_\tau \log \pi(a|b(.;\tau);\theta)$ is impossible to compute analytically
  - Φ is usually a handcrafted function which extracts non-continuous, very often binary features

Actor critic method

- Since we can compute

$$\nabla_\theta \log \pi(a | b(.;\tau);\theta)$$

- We can derive an algorithm for updating the policy parameters θ

- This results in

$$\nabla_\theta J(\theta,\tau) \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T_n} \nabla_\theta \log \pi(a_t^n | b(h_t^n;\tau);\theta) R(H_t^n)$$

Expected reward approximation

- $R(H_t^n)$ is an expectation over all rewards for H_t^n
  - which is not normally available and we have to compute it

- $R(H_t^n)$ can be approximated by a function which is compatible with the distribution $\pi(a|b(.;\tau);\theta)$

$$R(H^n) \approx R(H^n; w) = \sum_{t=0}^{T_n} \nabla_\theta \log \pi(a_t^n | b(h_t^n; \tau); \theta)^T w + C$$

  - the approximation cannot be arbitrary since the approximation can introduce some bias into the estimate gradient

Expected reward approximation

- To compute the expected reward
- Least squares can be used
  - replace $R\left(H_t^n\right)$ by observed rewards r_n at the end of dialogues

$$r_n = \sum_{t=0}^{T} \nabla_\theta \log \pi\left(a_t^n \middle| b\left(h_t^n;\tau\right);\theta\right)^T w + C \quad \forall\, n \in \{1, \ldots, N\}$$

Actor critic algorithm

- Having the approximation of $R\left(H_t^n\right)$
- We can construct a final gradient

$$\nabla_\theta J(\theta,\tau) \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T_n} \nabla_\theta \log \pi\left(a_t^n \middle| b\left(h_t^n;\tau\right);\theta\right) \left(\nabla_\theta \log \pi\left(a \middle| b\left(h_t^n;\tau\right);\theta\right)^T w + C\right)$$

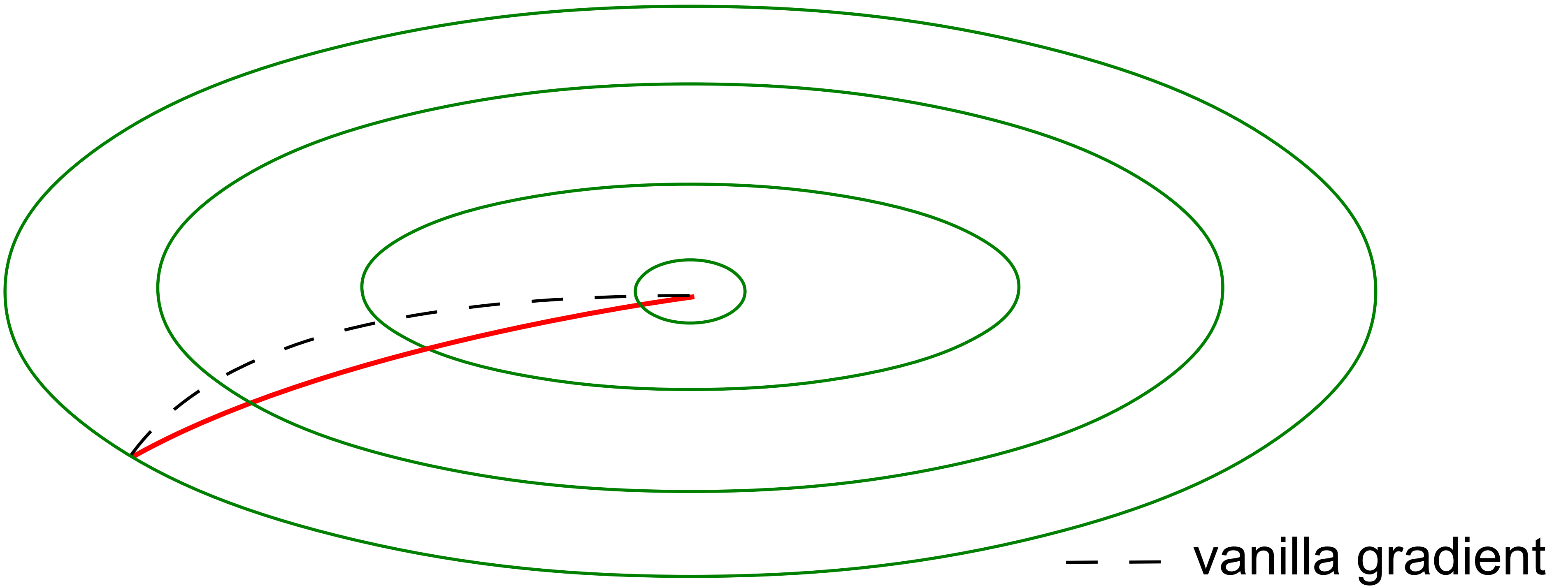
- Interestingly, if you ignore the constant C you get faster convergence
  - removing it lowers the variance of the gradient

Natural gradient

- Geometry of the parameter space is important

$$\tilde{\nabla}_{\theta} J(\theta,\tau) = F(\theta)^{-1} \nabla J(\theta,\tau)$$

Fisher information matrix



Natural Actor Critic

- Uses natural gradient to update the policy parameters

$$\tilde{\nabla}_\theta J(\theta, \tau) = F(\theta)^{-1} \nabla J(\theta, \tau)$$

- After substitution

$$\tilde{\nabla}_\theta J(\theta, \tau) \approx F(\theta)^{-1} \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T_n} \nabla_\theta \log \pi(a_t^n | b(h_t^n; \tau); \theta)$$
$$\nabla_\theta \log \pi(a | b(h_t^n; \tau); \theta)^T w$$

Natural Actor Critic

$$\tilde{\nabla}_\theta J(\theta,\tau) \approx F(\theta)^{-1} \boxed{\frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T_n} \nabla_\theta \log \pi(a_t^n | b(h_t^n;\tau);\theta) \nabla_\theta \log \pi(a | b(h_t^n;\tau);\theta)^T} w$$

One can notice that this is an estimate of the Fisher information matrix

- Therefore, it holds true

$$\tilde{\nabla}_\theta J(\theta,\tau) = F(\theta)^{-1} \nabla J(\theta,\tau) \approx F(\theta)^{-1} F(\theta) w = w$$

Natural Actor Critic

- The Fisher matrix does not have to be computed
- The core of the NAC method uses least squares algorithm to solve

$$r_n = \sum_{t=0}^{T_n} \nabla_\theta \log \pi\left(a_t^n | b_t^n, \theta\right)^T \cdot w_\theta + C \qquad \forall\, n \in 1, \ldots, N$$

- Update the parameters

$$\theta' = \theta + \beta_\theta w_\theta$$

NAC: Summary

- Natural Actor Critic method is an iterative algorithm
  - Sample some dialogues
  - Collect relevant statistics
    - observations, actions, rewards
  - Compute the natural gradient

$$r_n = \sum_{t=0}^{T_n} \nabla_\theta \log \pi\left(a_t^n \middle| b_t^n, \theta\right)^T \cdot w_\theta + C \qquad \forall\, n \in 1, \ldots, N$$

- Using natural gradient is orders of magnitude more efficient

- NAC needs much more dialogues than GP-SARSA

Thank you!

Filip Jurčíček

Institute of Formal and Applied Linguistics
Charles University in Prague
Czech Republic

Home page: http://ufal.mff.cuni.cz/~jurcicek

User simulation

- Ideally the POMDP dialogue systems would be optimised in interaction with real users
- The problem
  - state-of-the-art techniques still needs to more than 10000 dialogues
- User simulators are used to train and test POMDP techniques
- User simulators are mostly hand-crafted, though parametrised

User simulators

- Mostly implemented on the dialogue act level
  - S: request(food_type)
  - US: inform(food_type=Chinese)

- Many different implementations
  - bigram model for dialogue act types and random sampling slots
  - agenda based simulator
  - using user model to sample user dialogue acts