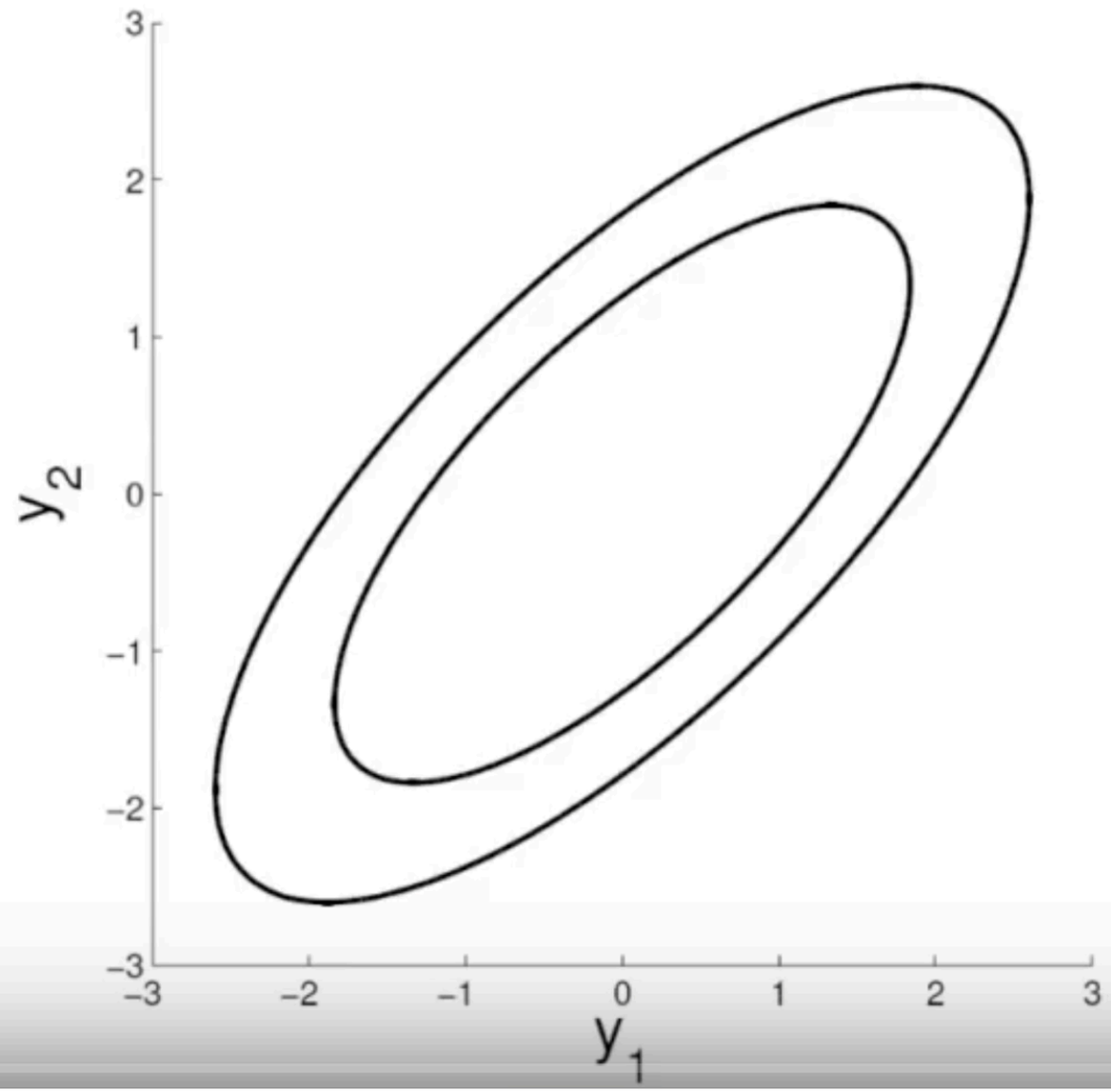
Gaussian Process Reinforcement Learning for Dialog Management

Maryam Zare

penn state university

Multivariate Gaussian Distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^{\mathsf{T}}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .7 \\ .7 & 1 \end{bmatrix}$$

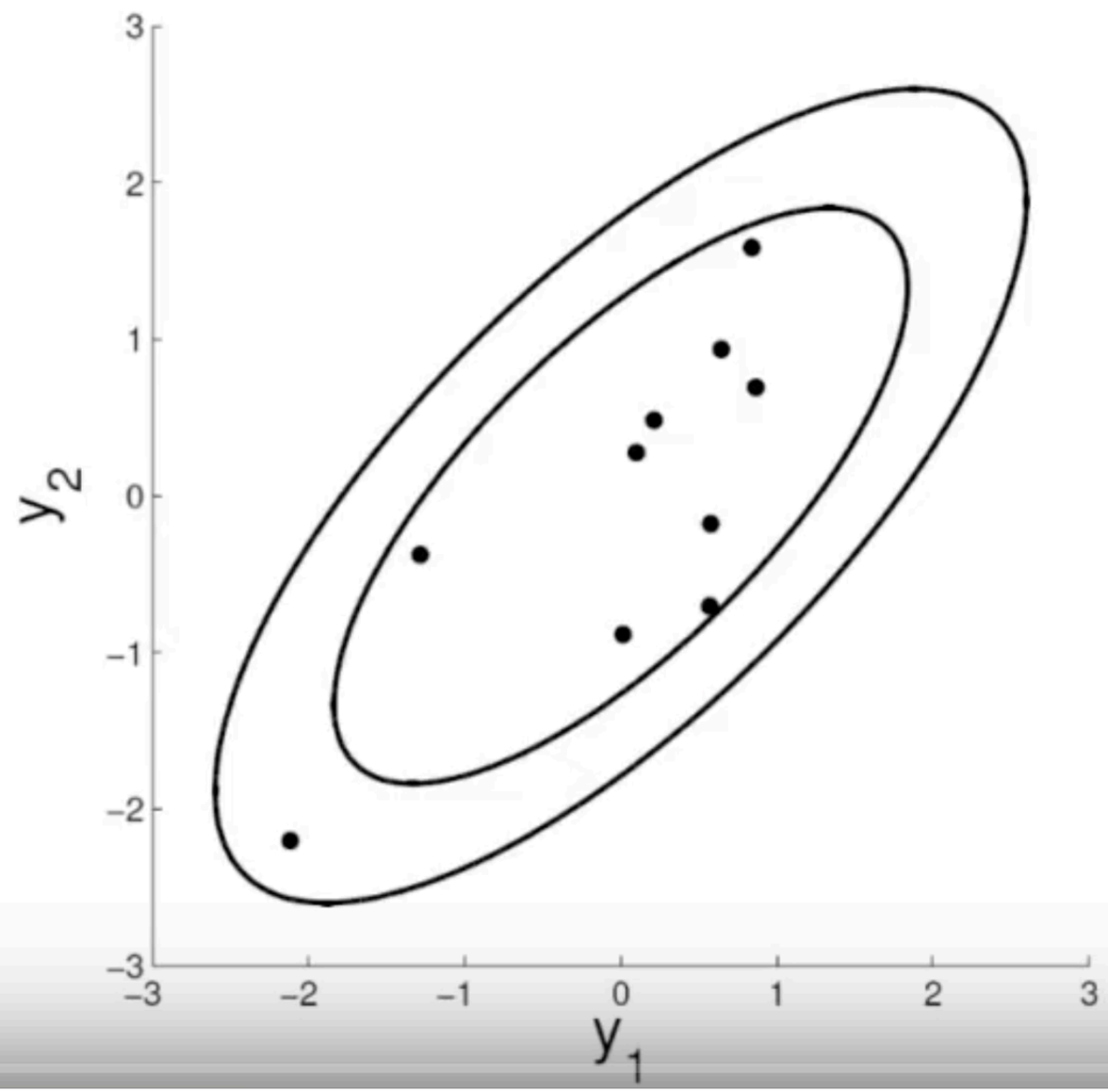


Examples are from: https://www.youtube.com/watch?v=92-98SYOdIY

Multivariate Gaussian Distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^{\mathsf{T}}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .7 \\ .7 & 1 \end{bmatrix}$$

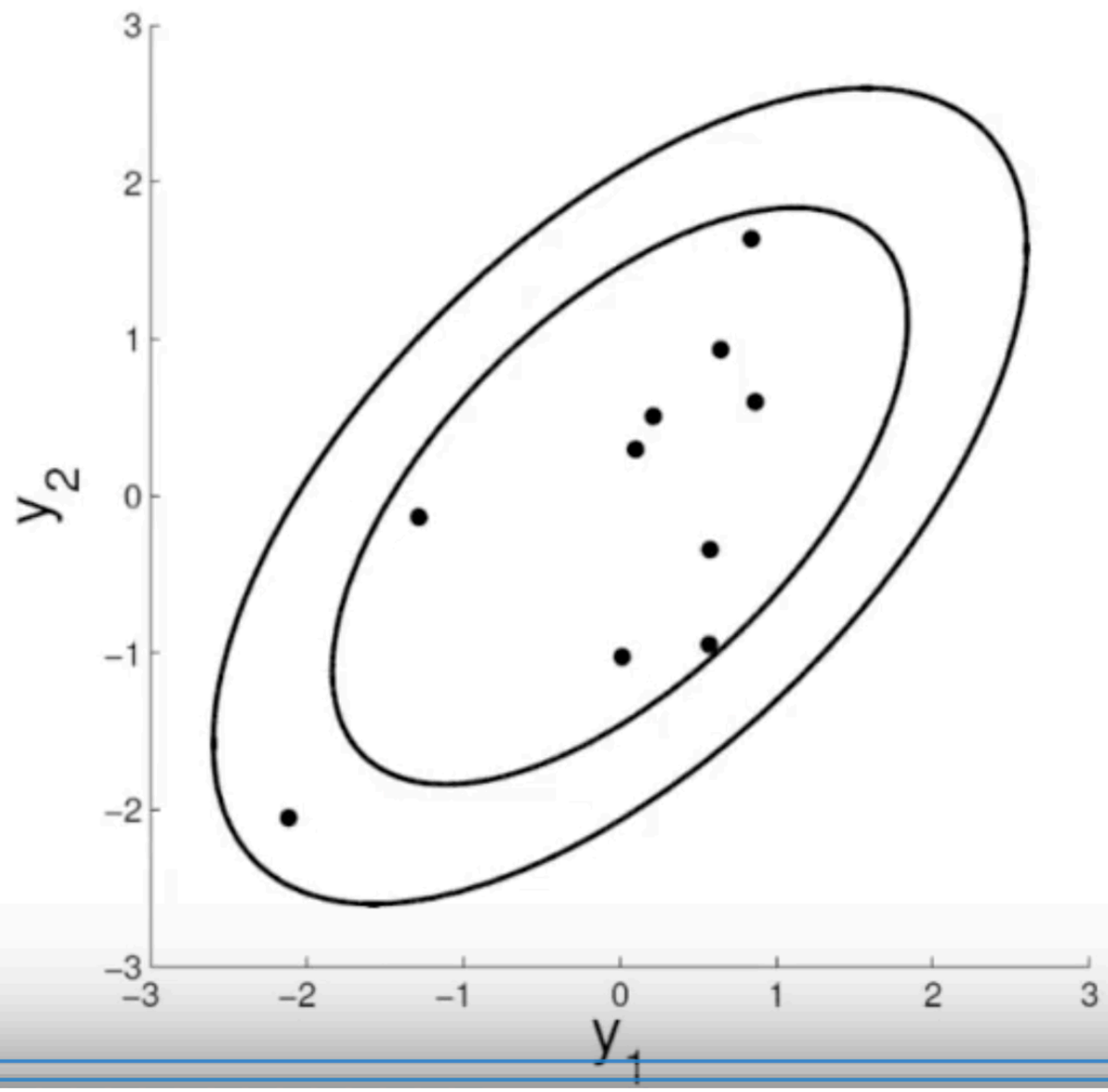
Y1 and Y2 are Highly Correlated



Examples are from: https://www.youtube.com/watch?v=92-98SYOdlY

Multivariate Gaussian Distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^{\mathsf{T}}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .6 \\ .6 & 1 \end{bmatrix}$$

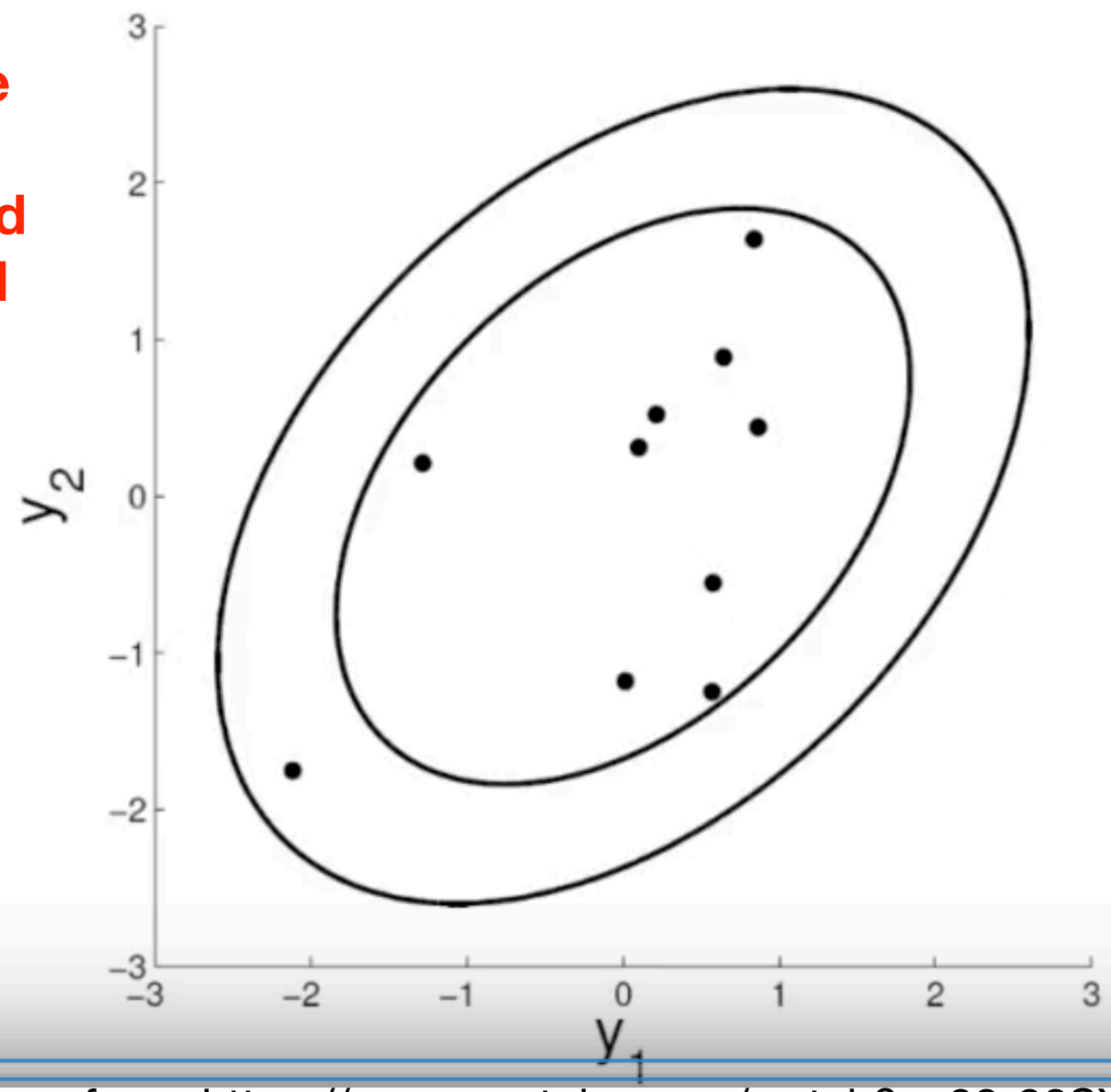


Examples are from: https://www.youtube.com/watch?v=92-98SYOdIY

Multivariate Gaussian Distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^{\mathsf{T}}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .4 \\ .4 & 1 \end{bmatrix}$$

Samples are Less Concentrated On Diagonal

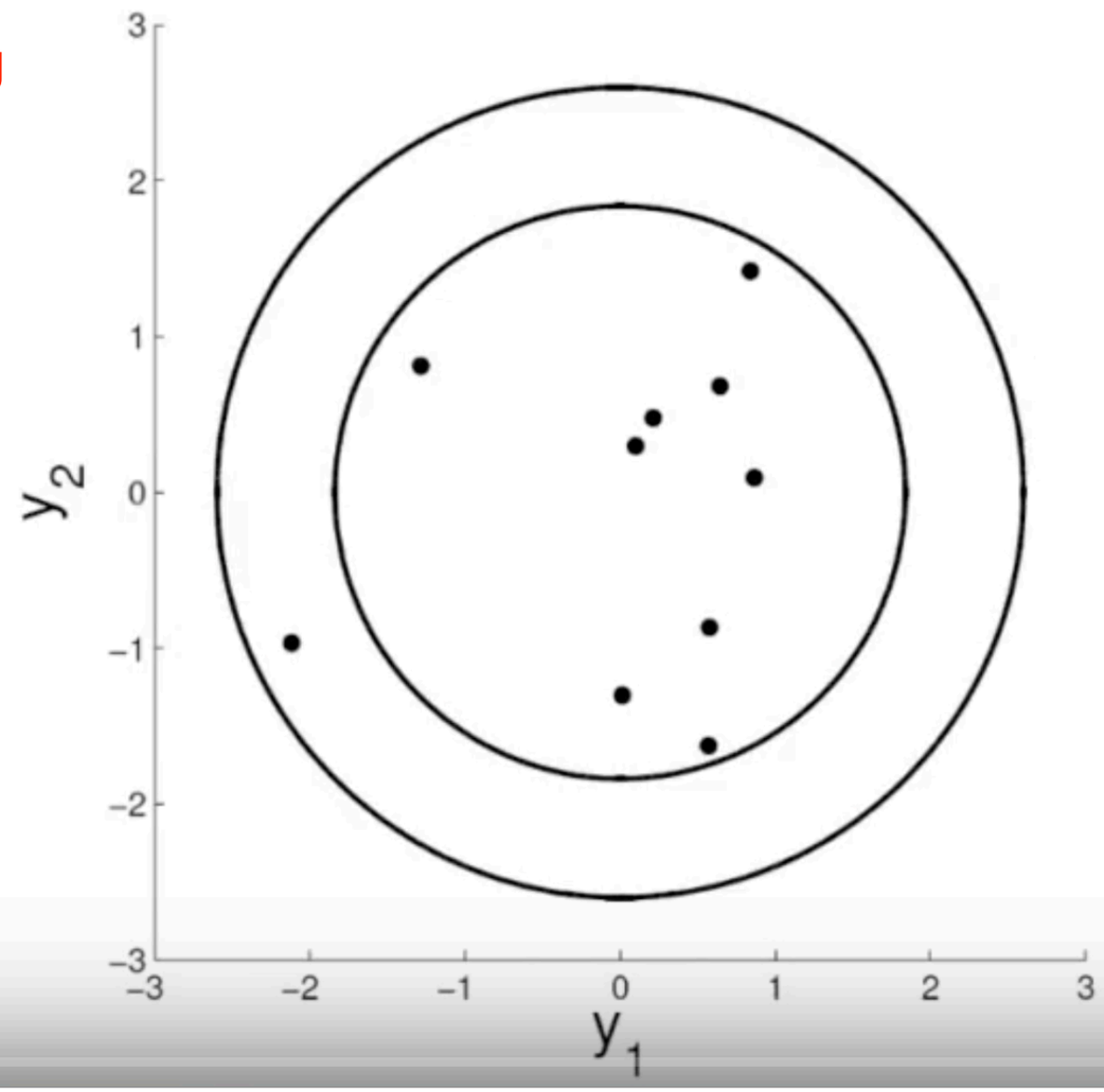


Examples are from: https://www.youtube.com/watch?v=92-98SYOdIY

Multivariate Gaussian Distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^{\mathsf{T}}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

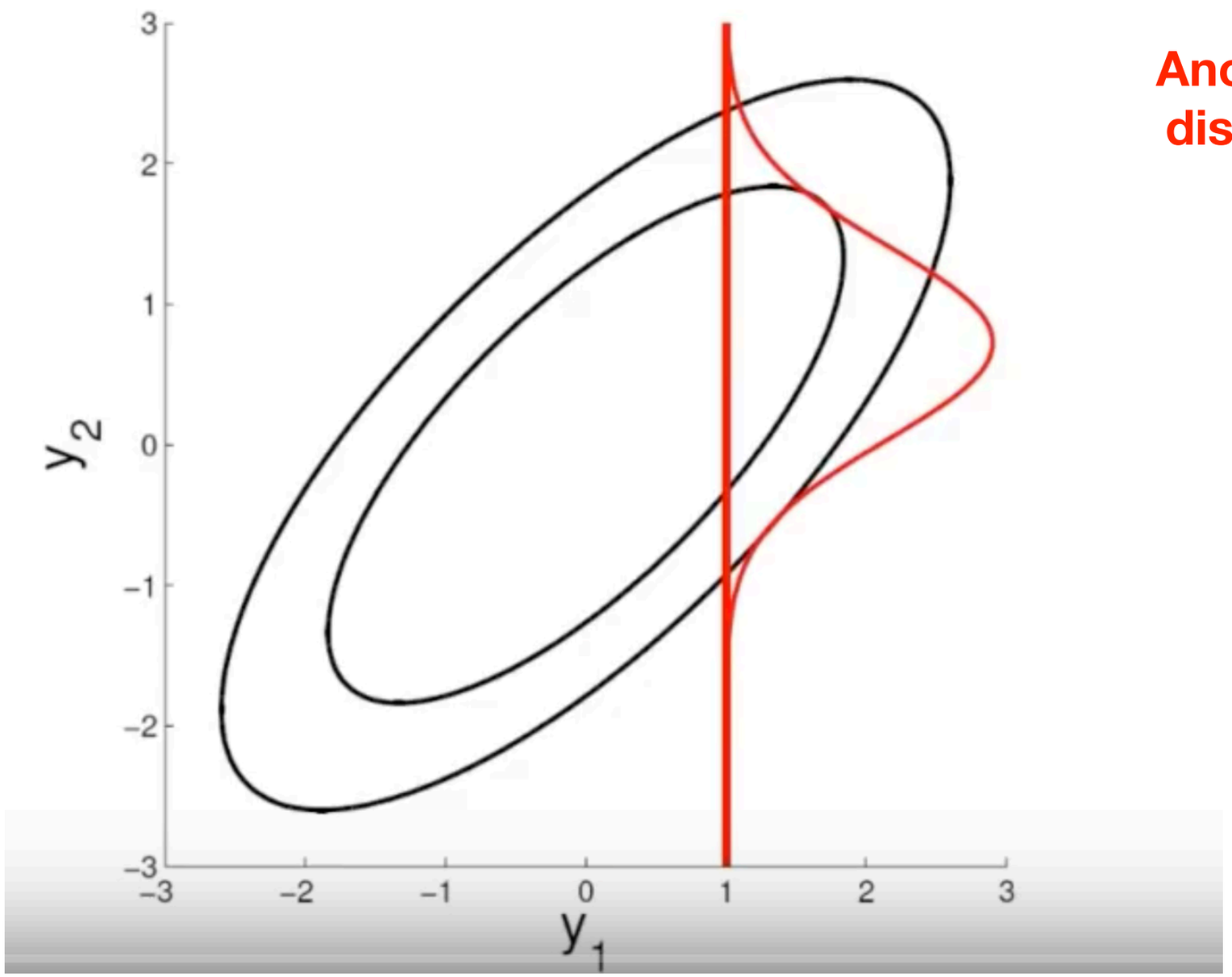
Y2 tells nothing about Y1



Examples are from: https://www.youtube.com/watch?v=92-98SYOdlY

Conditioning Gaussian Distribution

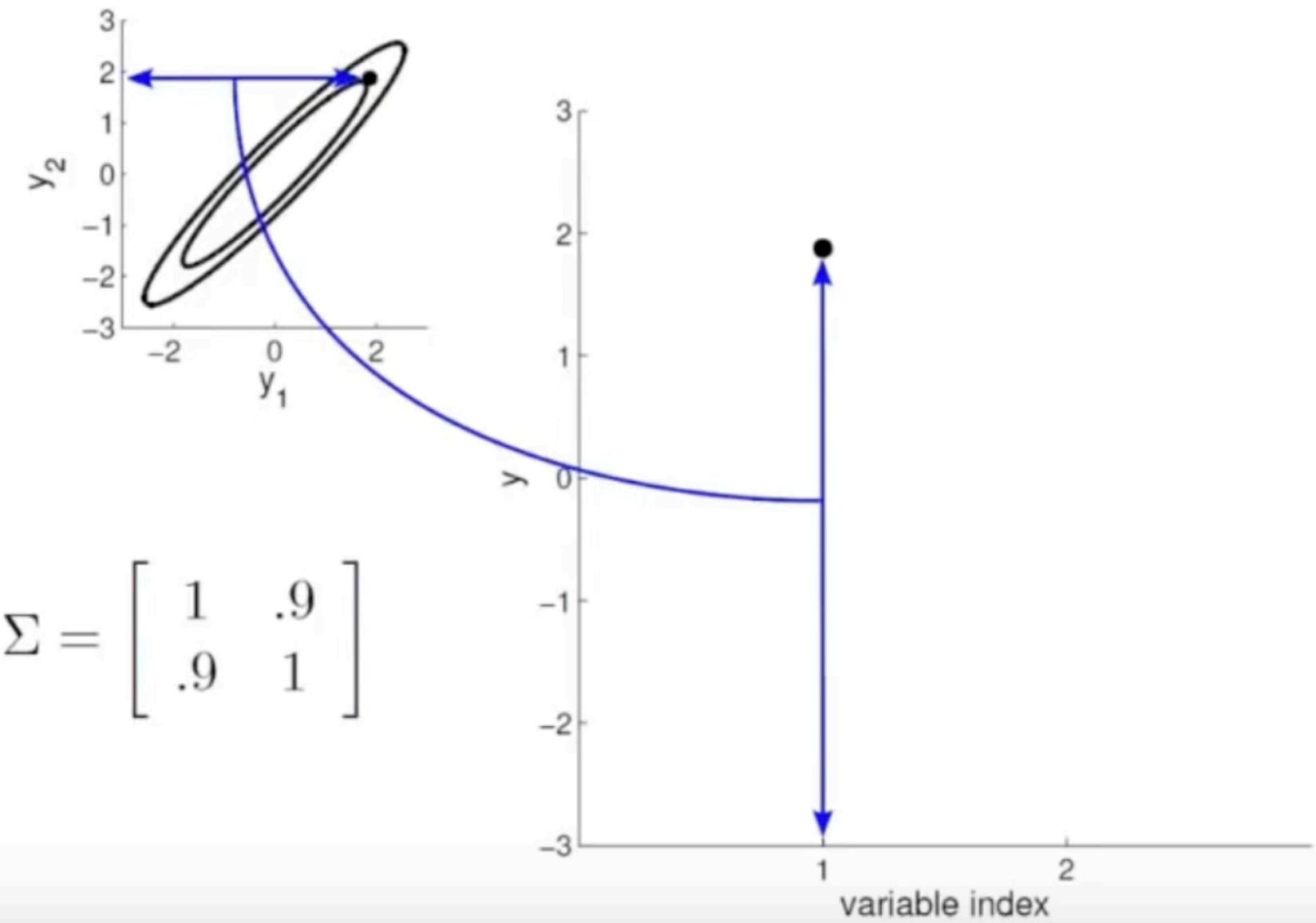
$$p(y_2|y_1, \Sigma) \propto \exp\left(-\frac{1}{2}(y_2 - \mu_*)\Sigma_*^{-1}(y_2 - \mu_*)\right)$$



Another Gaussian distribution in the Variable Y2

Examples are from: https://www.youtube.com/watch?v=92-98SYOdIY

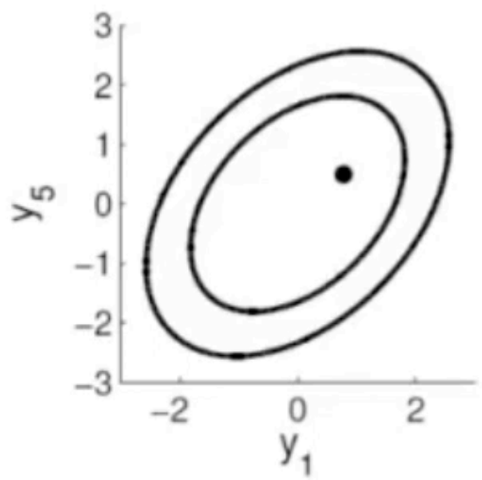
A new visualization

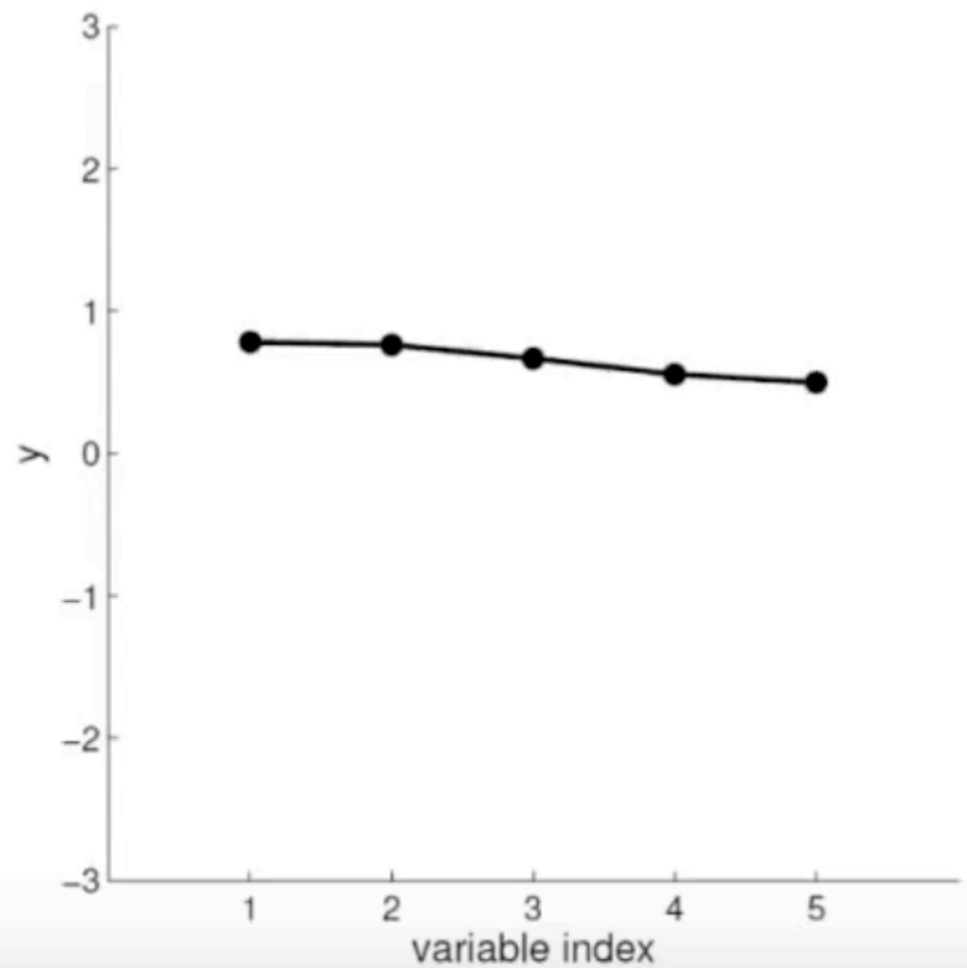


$$\Sigma = \begin{bmatrix} 1 & .9 \\ .9 & 1 \end{bmatrix}$$

Examples are from: https://www.youtube.com/watch?v=92-98SYOdIY

A new visualization

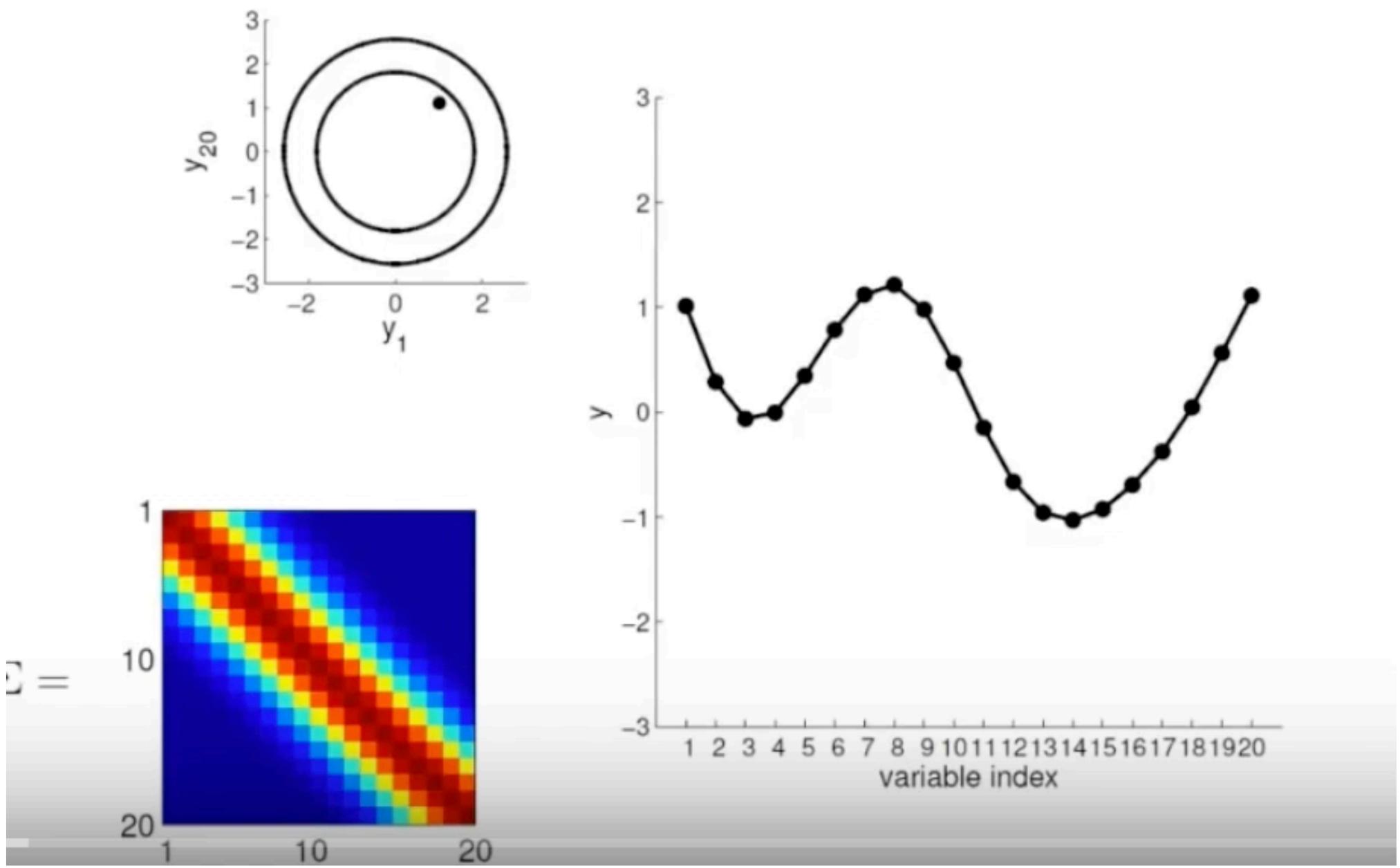




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

Examples are from: https://www.youtube.com/watch?v=92-98SYOdlY

A new visualization (infinite variables)



Examples are from: https://www.youtube.com/watch?v=92-98SYOdlY

Gaussian Process

Non-parametric (∞-parametric)

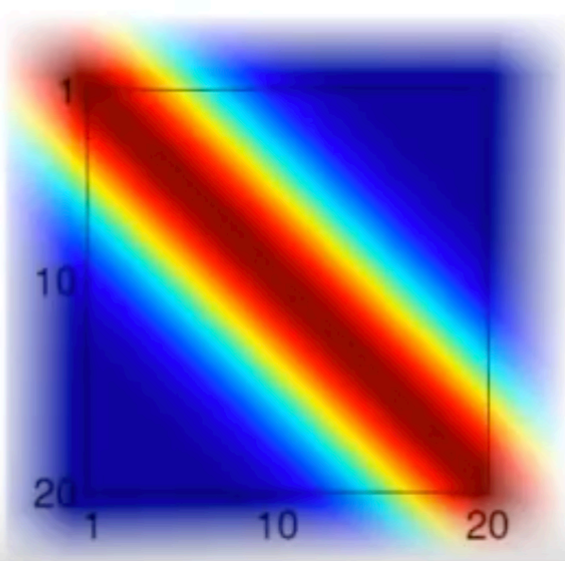
$$p(\mathrm{y}|\theta) = \mathcal{N}(0, \Sigma)$$

$$\Sigma(\mathrm{x}_1, \mathrm{x}_2) = \mathrm{K}(\mathrm{x}_1, \mathrm{x}_2) + \mathrm{I}\sigma_{\mathrm{y}}^2$$

$$\mathrm{K}(\mathrm{x}_1, \mathrm{x}_2) = \sigma^2 \mathrm{e}^{-\frac{1}{2\mathrm{l}^2}(\mathrm{x}_1 - \mathrm{x}_2)^2}$$

vertical-scale horizontal-scale

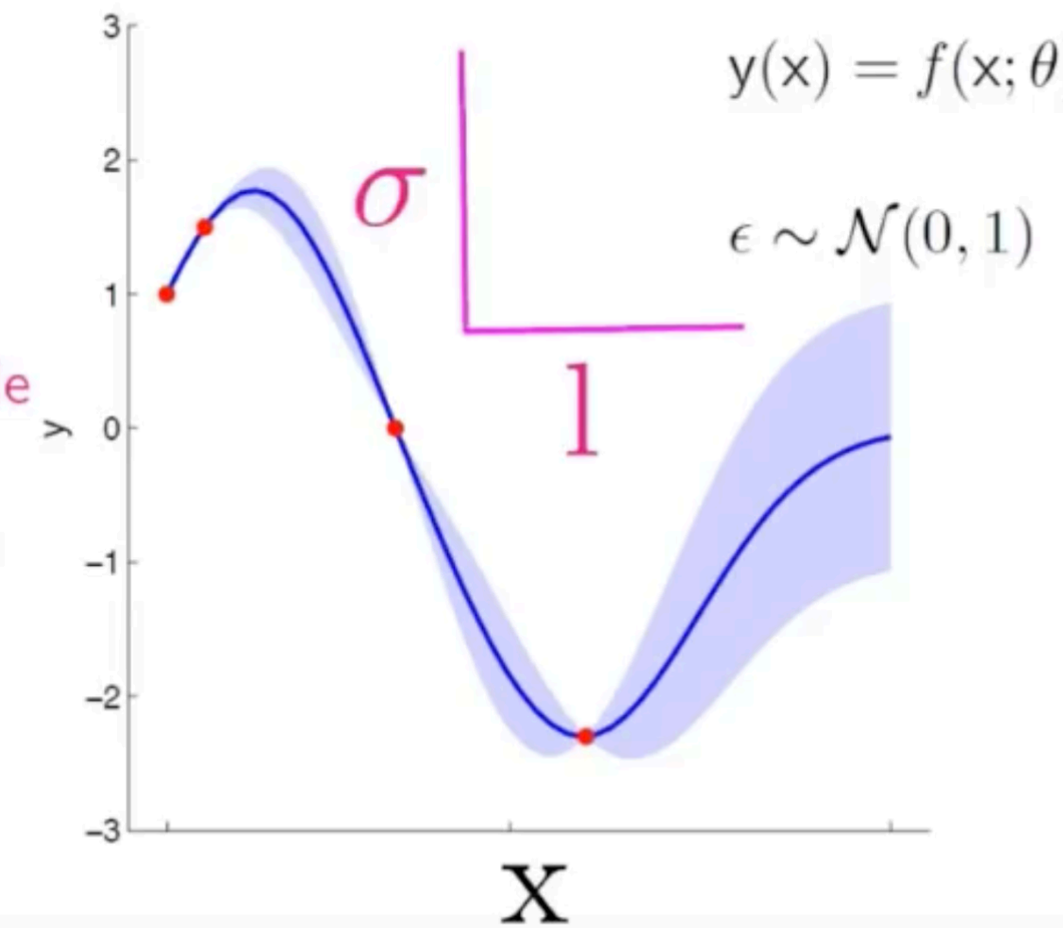
$\Sigma =$



Parametric model

$$\mathrm{y}(\mathrm{x}) = f(\mathrm{x}; \theta) + \sigma_{\mathrm{y}}\epsilon$$

$$\epsilon \sim \mathcal{N}(0, 1)$$



Examples are from: https://www.youtube.com/watch?v=92-98SYOdlY

A Gaussian Process Model for the Q Function

- The discounted reward for time step ***t***, given policy ***π***:

$$R_t^{\pi} = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}, \qquad (1)$$

Discounted Factor [0,1]

A Gaussian Process Model for the Q Function

- The discounted reward for time step ***t***, given policy ***π***:

$$R_t^\pi = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}, \qquad (1)$$

- We can write the discounted return recursively as

$$R_t^\pi = r_{t+1} + \gamma R_{t+1}^\pi. \qquad (2)$$

Immediate Reward **Future Rewards**

A Gaussian Process Model for the Q Function

- The discounted reward for time step ***t***, given policy ***π***:

$$R_t^\pi = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}, \qquad (1)$$

- We can write the discounted return recursively as

$$R_t^\pi = r_{t+1} + \gamma R_{t+1}^\pi. \qquad (2)$$

- The Q-function for policy ***π*** given belief state ***b*** and action ***a*** at time ***t*** is

$$Q^\pi(\mathbf{b}, a) = E_\pi \left(R_t | b(s_t) = \mathbf{b}, a_t = a \right). \qquad (3)$$

expectation of the discounted return given belief state *b* and action *a* at time *t*, over all possible belief state sequences that can be generated

A Gaussian Process Model for the Q Function

- Transitions are stochastic, therefore:

$$R_t^\pi(b(s_t) = \mathbf{b}, a_t = a) = Q^\pi(\mathbf{b}, a) + \Delta Q^\pi(\mathbf{b}, a). \quad (4)$$

Random Variable **Mean** **Residual**

A Gaussian Process Model for the Q Function

- Transitions are stochastic, therefore:

$$R_t^\pi(b(s_t) = \mathbf{b}, a_t = a) = Q^\pi(\mathbf{b}, a) + \Delta Q^\pi(\mathbf{b}, a). \quad (4)$$

- Substituting return at time step **t** and **t+1** from (4) into (2) yields:

$$\begin{aligned} r_{t+1}(b(s_t) = \mathbf{b}, a_t = a) \quad &= Q^\pi(\mathbf{b}, a) - \gamma Q^\pi(\mathbf{b}', a') \\ &+ \Delta Q^\pi(\mathbf{b}, a) - \gamma \Delta Q^\pi(\mathbf{b}', a'), \end{aligned} \quad (5)$$

A Gaussian Process Model for the Q Function

- Transitions are stochastic, therefore:

$$R_t^\pi(b(s_t) = \mathbf{b}, a_t = a) = Q^\pi(\mathbf{b}, a) + \Delta Q^\pi(\mathbf{b}, a). \quad (4)$$

- Substituting return at time step **t** and **t+1** from (4) into (2) yields:

$$\begin{aligned} r_{t+1}(b(s_t) = \mathbf{b}, a_t = a) \quad &= Q^\pi(\mathbf{b}, a) - \gamma Q^\pi(\mathbf{b}', a') \\ &+ \Delta Q^\pi(\mathbf{b}, a) - \gamma \Delta Q^\pi(\mathbf{b}', a'), \end{aligned} \quad (5)$$

- In general we can write (5) as

$$\begin{aligned} r^1 \quad &= Q^\pi(\mathbf{b}^0, a^0) - \gamma Q^\pi(\mathbf{b}^1, a^1) \\ &+ \Delta Q^\pi(\mathbf{b}^0, a^0) - \gamma \Delta Q^\pi(\mathbf{b}^1, a^1) \\ r^2 \quad &= Q^\pi(\mathbf{b}^1, a^1) - \gamma Q^\pi(\mathbf{b}^2, a^2) \\ &+ \Delta Q^\pi(\mathbf{b}^1, a^1) - \gamma \Delta Q^\pi(\mathbf{b}^2, a^2) \\ &\vdots \\ r^t \quad &= Q^\pi(\mathbf{b}^{t-1}, a^{t-1}) - \gamma Q^\pi(\mathbf{b}^t, a^t) \\ &+ \Delta Q^\pi(\mathbf{b}^{t-1}, a^{t-1}) - \gamma \Delta Q^\pi(\mathbf{b}^t, a^t), \end{aligned} \quad (6)$$

A Gaussian Process Model for the Q Function

- Or in a **compact** form we have

$$\mathbf{r}_t = \mathbf{H}_t \mathbf{q}_t^\pi + \mathbf{H}_t \boldsymbol{\Delta} \mathbf{q}_t^\pi, \tag{7a}$$

- Where,

$$\mathbf{r}_t = [r^1, \ldots, r^t]^\top \tag{7b}$$

$$\mathbf{q}_t^\pi = [Q^\pi(\mathbf{b}^0, a^0), \ldots, Q^\pi(\mathbf{b}^t, a^t)]^\top, \tag{7c}$$

$$\boldsymbol{\Delta} \mathbf{q}_t^\pi = [\Delta Q^\pi(\mathbf{b}^0, a^0), \ldots, \Delta Q^\pi(\mathbf{b}^t, a^t)]^\top, \tag{7d}$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -\gamma \end{bmatrix}. \tag{7e}$$

A Gaussian Process Model for the Q Function

- Or in a **compact** form we have

$$\mathbf{r}_t = \mathbf{H}_t \mathbf{q}_t^\pi + \mathbf{H}_t \boldsymbol{\Delta} \mathbf{q}_t^\pi, \tag{7a}$$

- Where,

$$\mathbf{r}_t = [r^1, \ldots, r^t]^\top \tag{7b}$$

$$\mathbf{q}_t^\pi = [Q^\pi(\mathbf{b}^0, a^0), \ldots, Q^\pi(\mathbf{b}^t, a^t)]^\top, \tag{7c}$$

$$\boldsymbol{\Delta} \mathbf{q}_t^\pi = [\Delta Q^\pi(\mathbf{b}^0, a^0), \ldots, \Delta Q^\pi(\mathbf{b}^t, a^t)]^\top, \tag{7d}$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -\gamma \end{bmatrix}. \tag{7e}$$

A Gaussian Process Model for the Q Function

Transition probabilities and policy π change over time

→ Q(b, a) is a random variable

→ We can model it as a Gaussian Process:

$$Q^{\pi}(\mathbf{b}, a) \sim \mathcal{GP}\left(0, k((\mathbf{b}, a), (\mathbf{b}, a))\right),$$

Where the kernel k(·,·) is factored into separate kernels over the belief state and action space:

$$k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}')k_{\mathcal{A}}(a, a')$$

And the Gaussian noise is

$$\Delta Q^{\pi}(\mathbf{b}, a) \sim \mathcal{N}(0, \sigma^2)$$

Gaussian Process Based Policy Model

- How to define the policy?
  - The Simplest way

$$\pi(\mathbf{b}) = \arg\max_{a} \overline{Q}(\mathbf{b}, a). \tag{10}$$

select the action associated with the highest mean of the Q-function

Gaussian Process Based Policy Model

- How to define the policy?
  - The Simplest way

$$\pi(\mathbf{b}) = \arg\max_a \overline{Q}(\mathbf{b}, a). \tag{10}$$

  - A better way (epsilon greedy)

$$\pi(\mathbf{b}) = \begin{cases} \arg\max_a \overline{Q}(\mathbf{b}, a) & \text{with prob } 1-\epsilon, \\ \text{random } a & \text{with prob } \epsilon. \end{cases} \tag{11}$$

parameter ε which balances how often an action is taken according to the current best estimate of the Q-function mean (the *exploitation* stage) compared to how often an action is taken randomly (the *exploration* stage)

Gaussian Process Based Policy Model

- How to define the policy?
  - The Simplest way

$$\pi(\mathbf{b}) = \arg\max_a \overline{Q}(\mathbf{b}, a). \tag{10}$$

  - A better way (epsilon greedy)

$$\pi(\mathbf{b}) = \begin{cases} \arg\max_a \overline{Q}(\mathbf{b}, a) & \text{with prob } 1-\epsilon, \\ \text{random } a & \text{with prob } \epsilon. \end{cases} \tag{11}$$

  - Even better (explore the part of space where we're less certain about)

$$\pi(\mathbf{b}) = \begin{cases} \arg\max_a \overline{Q}(\mathbf{b}, a) & \text{with prob } 1-\epsilon, \\ \arg\max_a \operatorname{cov}\left((\mathbf{b}, a), (\mathbf{b}, a)\right) & \text{with prob } \epsilon. \end{cases} \tag{12}$$

Gaussian Process Based Policy Model

- How to define the policy?
  - Or just sample!
    - Since the Gaussian process for the Q-function defines a Gaussian distribution for every belief state-action pair (8), when a new belief point ***b*** is encountered, for each action ***a*** $\in$ ***A***, there is a Gaussian distribution,

$$\hat{Q}(\mathbf{b}, a) \sim \mathcal{N}(\overline{Q}(\mathbf{b}, a), \mathrm{cov}((\mathbf{b}, a), (\mathbf{b}, a))).$$

    - Sampling from these Gaussian distributions gives Q-values from which the action with the highest sampled Q-value can be selected

$$\pi(\mathbf{b}) = \arg\max_{a} \left\{ \hat{Q}(\mathbf{b}, a) : a \in \mathcal{A} \right\}. \qquad (13)$$

Gaussian Process-Based Policy Optimization

- Sarsa Algorithm

$$Q(\mathbf{b}, a) \leftarrow Q(\mathbf{b}, a) + \alpha[r_{t+1}(b_t = \mathbf{b}, a_t = a) + \gamma Q(\mathbf{b}', a') - Q(\mathbf{b}, a)]. \tag{14}$$

If the Q-function correction provided by the right hand side of (14) is replaced by the update of the GP-based Q-function posterior given by (8), the GP-Sarsa algorithm is obtained.

Gaussian Process-Based Policy Optimization

Fig. 1. Episodic GP-Sarsa (without computational constraints)

for each episode **do**
  Initialise $\mathbf{b}$
  if first episode **then**
    Choose a arbitrary
    $\mathbf{B}_0 \leftarrow (\mathbf{b}, a)$
    $\mathbf{K}_0 \leftarrow k((\mathbf{b}, a), (\mathbf{b}, a))$
    $\mathbf{H}_0 \leftarrow \begin{bmatrix} 1 & -\gamma \end{bmatrix}$
  else
    if initial step **then**
      Choose $a \leftarrow \pi(\mathbf{b})$ (11), (12) or (13)
    end if
  end if
  for each step in the episode **do**
    Take a, observe r', update $\mathbf{b}'$
    if non-terminal step **then**
      Choose new action $a' \leftarrow \pi(\mathbf{b}')$ (11), (12) or (13)
      $\mathbf{B}_{t+1} \leftarrow \begin{bmatrix} \mathbf{B}_t & (\mathbf{b}', a') \end{bmatrix}$
      $\mathbf{H}_{t+1} \leftarrow \begin{bmatrix} \mathbf{H}_t & \mathbf{0} \\ \mathbf{u}^\top & -\gamma \end{bmatrix}$, where $\mathbf{u} = \begin{bmatrix} \mathbf{0} & 1 \end{bmatrix}^\top$
      $\mathbf{K}_{t+1} \leftarrow \begin{bmatrix} \mathbf{K}_t & \mathbf{k}_t(\mathbf{b}', a') \\ \mathbf{k}_t(\mathbf{b}', a') & k(\mathbf{b}', a', \mathbf{b}', a') \end{bmatrix}$
    else
      $\mathbf{B}_{t+1} \leftarrow \mathbf{B}_t$, $\mathbf{K}_{t+1} \leftarrow \mathbf{K}_t$
      $\mathbf{H}_{t+1} \leftarrow \begin{bmatrix} \mathbf{H}_t \\ \mathbf{u}^\top \end{bmatrix}$, where $\mathbf{u} = \begin{bmatrix} \mathbf{0} & 1 \end{bmatrix}^\top$
    end if
    $\mathbf{r}_{t+1} \leftarrow \begin{bmatrix} \mathbf{r}_t & r' \end{bmatrix}$
    Update Q-function posterior $Q^\pi | \mathbf{r}_{t+1}, \mathbf{B}_{t+1}$ (8)
    if non-terminal step **then**
      $\mathbf{b} \leftarrow \mathbf{b}'$, $a \leftarrow a'$
    end if
  end for
end for

Choice of policy optimization

Update with GP-Sarsa

Experimental Set-up

Natural Actor Critic[1]:

belief space B is mapped into a summary space C. Likewise, the action space is mapped into a smaller scale summary action space A. This summary space C × A is then mapped into a feature space F . This space is then used to produce a parametric representation of the policy. A weighted set of these basis functions can then be cast into a probability via a soft-max function :

$$\pi(a|\mathbf{c};\theta) = \frac{e^{\theta \cdot f_a(\mathbf{c})}}{\sum_a e^{\theta \cdot f_a(\mathbf{c})}}. \qquad (19)$$

- Mappings B → C and C × A → F : require a significant amount of hand-crafting.
- the solution is only optimal within the given basis.
- gradient-based optimization methods are inherently slow which prevents direct on-line optimization with real users.

1. J.PetersandS.Schaal,“NaturalActor-Critic,”*Neurocomputing*,vol.71, pp. 1180–1190, 2008.

Experimental Set-up

Natural Actor Critic[1]:

belief space B is mapped into a summary space C. Likewise, the action space is mapped into a smaller scale summary action space A. This summary space C × A is then mapped into a feature space F . This space is then used to produce a parametric representation of the policy. A weighted set of these basis functions can then be cast into a probability via a soft-max function :

$$\pi(a|\mathbf{c};\theta) = \frac{e^{\theta \cdot f_a(\mathbf{c})}}{\sum_a e^{\theta \cdot f_a(\mathbf{c})}}. \qquad (19)$$

- Mappings B → C and C × A → F : require a significant amount of hand-crafting.
- the solution is only optimal within the given basis.
- gradient-based optimization methods are inherently slow which prevents direct on-line optimization with real users.

GP-Sarsa

The noise parameter σ and the sparsification parameter ν need to be defined. Whilst the choice of the kernel function and the choice of σ do not greatly affect the resulting policy, ***they do influence the speed of learning.*** In addition, the parameter ν controls the approximation of the kernel function.

- Kernel function choices
  - Polynomial

$$k(\mathbf{c}, \mathbf{c}'; \sigma_0, p) = (\langle \mathbf{c}, \mathbf{c}' \rangle + \sigma_0^2)^p, \qquad (21)$$

  - Gaussian

$$k(\mathbf{c}, \mathbf{c}'; p, \sigma_k) = p^2 \exp\left(- \frac{\|\mathbf{c} - \mathbf{c}'\|^2}{2\sigma_k^2} \right), \qquad (22)$$

- Policy Model

$$Q(\mathbf{b}, a) \sim \mathcal{N}(\overline{Q}(\mathbf{b}, a), \eta^2 \mathrm{cov}((\mathbf{b}, a), (\mathbf{b}, a))))$$

for a stochastic policy with variances scaled by η = 1, η = 2 and η = 3.

1. J.PetersandS.Schaal,“NaturalActor-Critic,”*Neurocomputing*,vol.71, pp. 1180–1190, 2008.

Results (GP-Sarsa vs NAC)

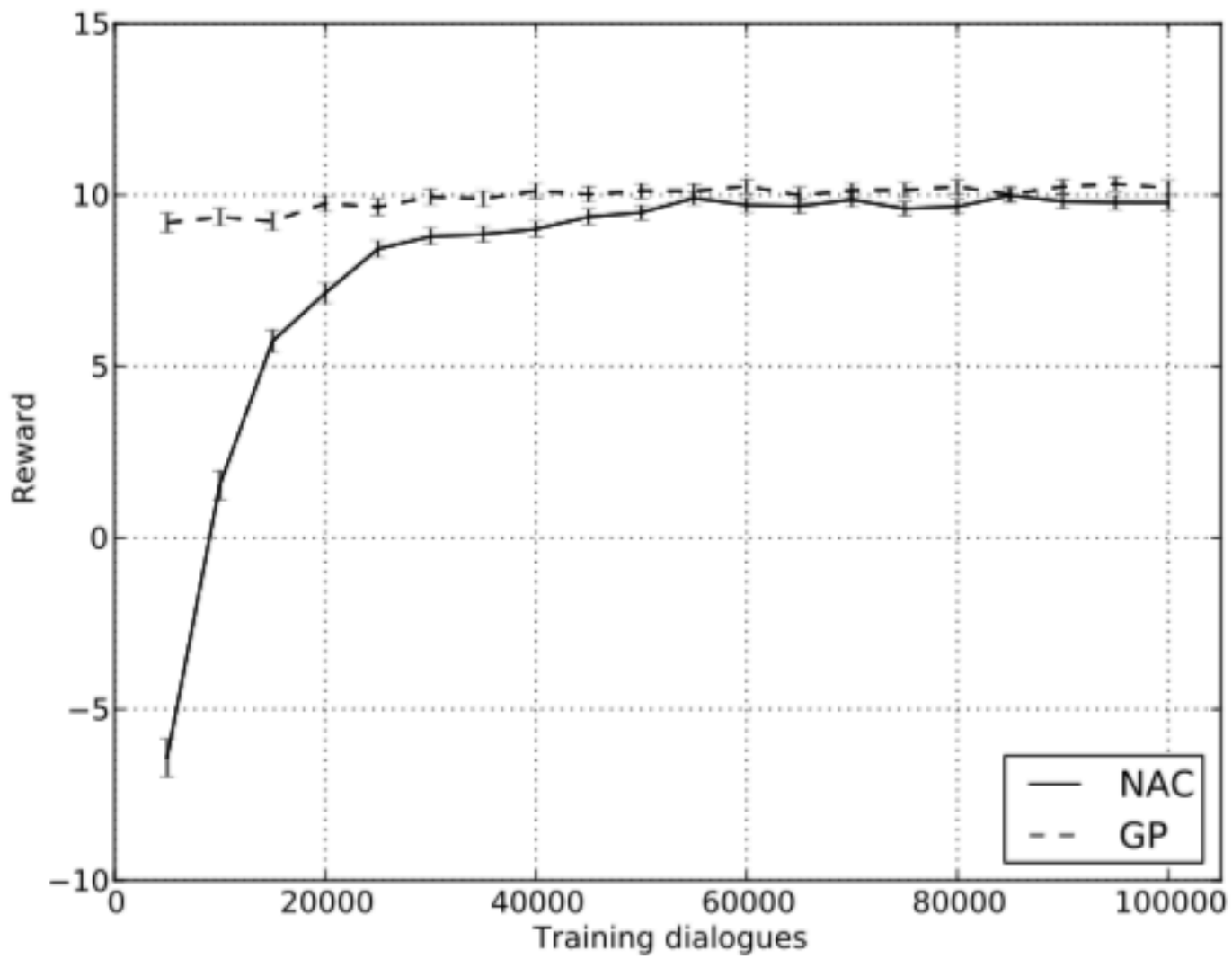


Fig. 9. Comparison of the performance of GP-Sarsa vs the natural actor critic algorithm, where the performance is measured as the average reward as a function of the number of training dialogues.

Results (GP-Sarsa vs NAC)

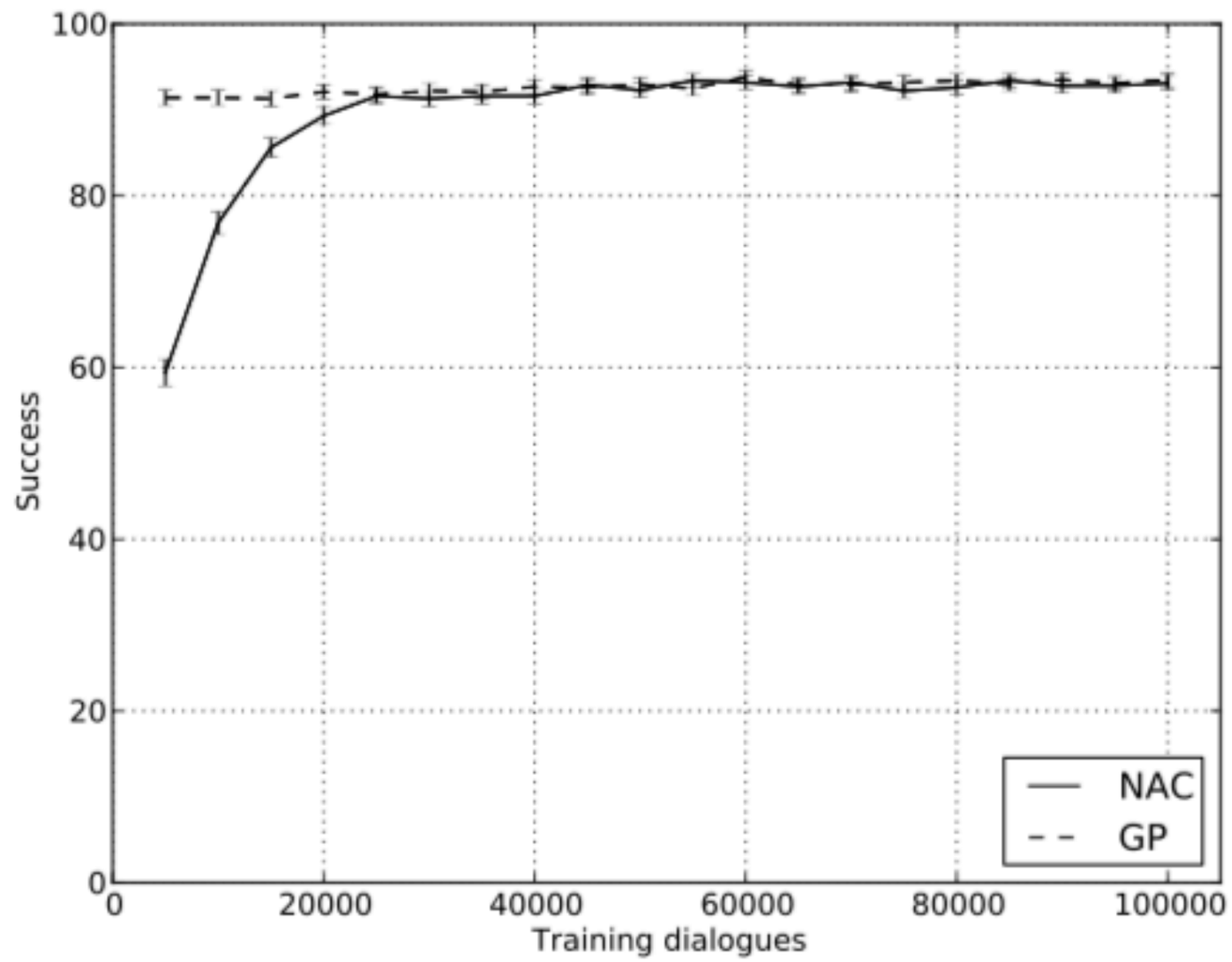


Fig. 10. Comparison of the performance of GP-Sarsa vs the natural actor critic algorithm, where the performance is measured as the average success as a function of the number of training dialogues.

Results (Influence of η During training)

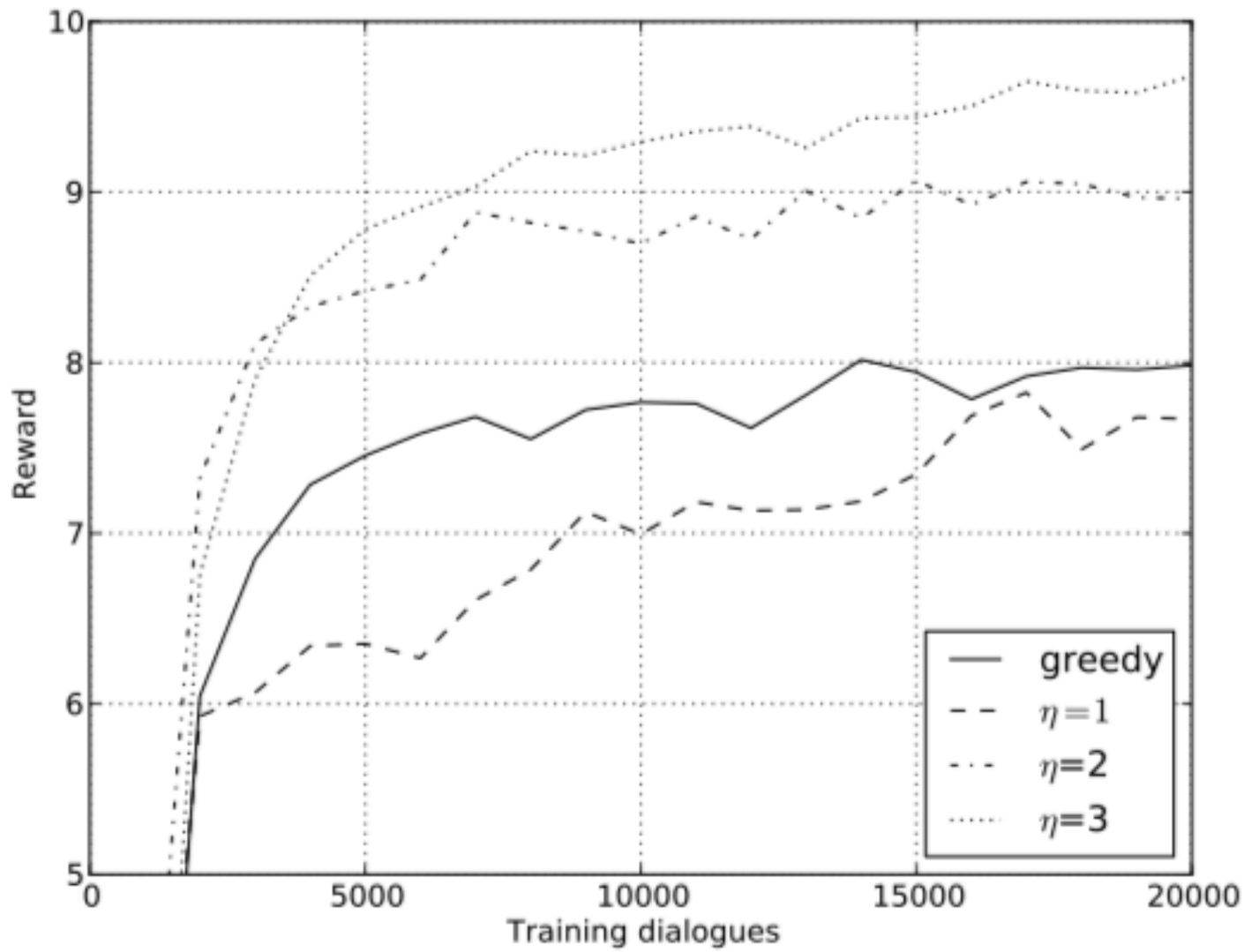


Fig. 7. Influence of variance scaling and the choice of the policy model on the performance during training.

Results (Influence of η During Testing)

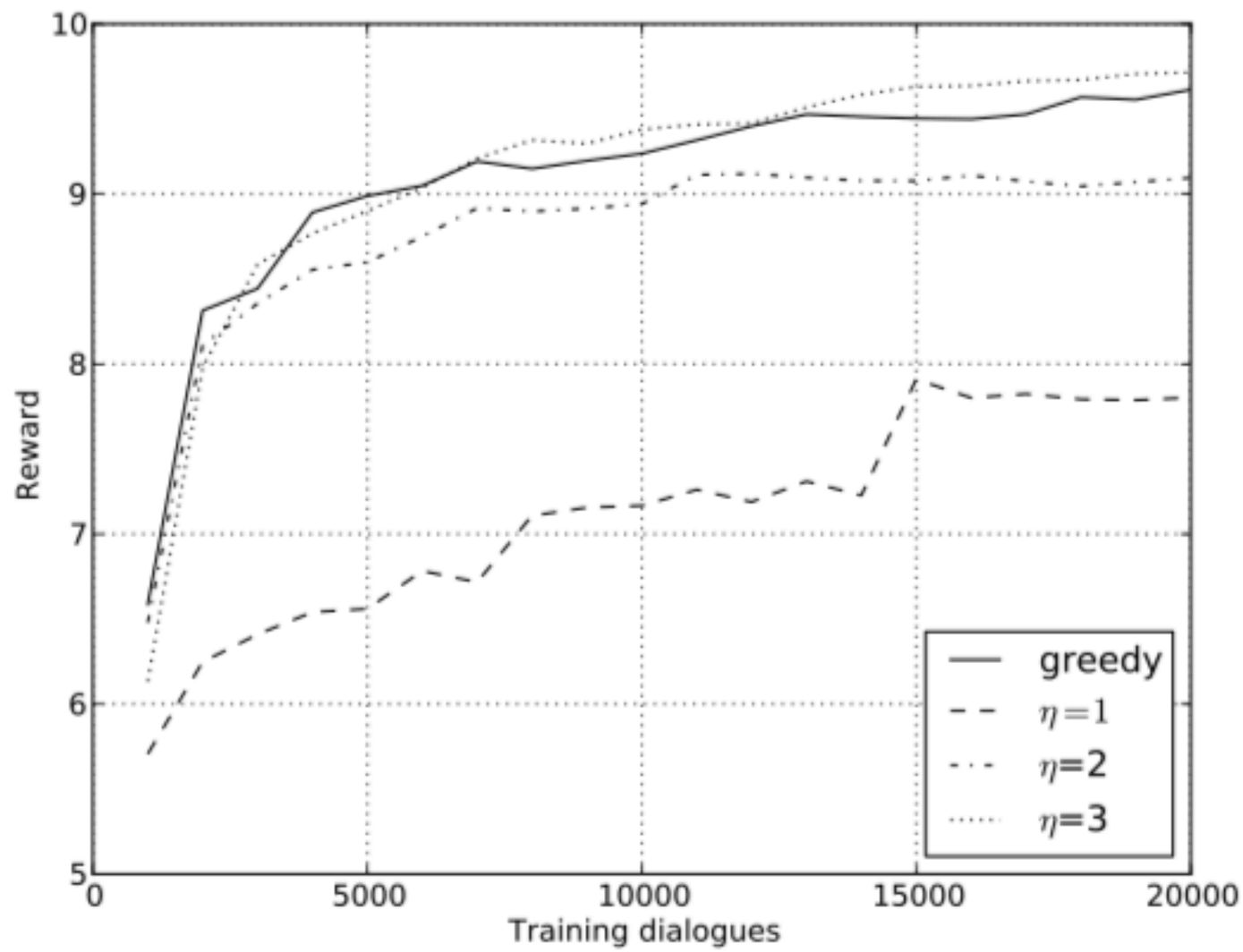


Fig. 8. Influence of variance scaling and the choice of the policy model on the performance during testing.

Results (Influence of Kernel)

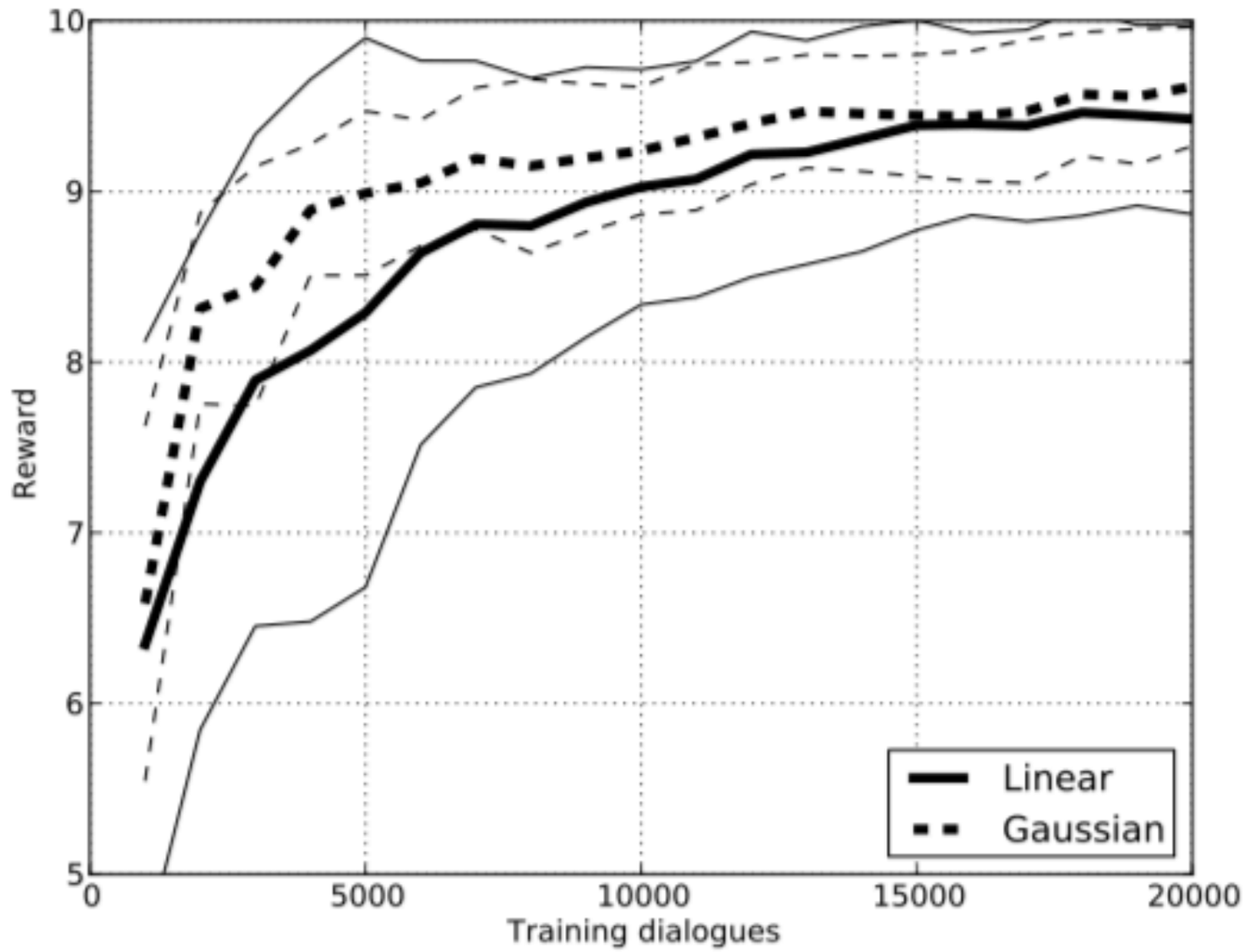


Fig. 3. Influence of the kernel choice on the performance. The performance is measured as the mean average reward as a function of the number of training dialogues. The thinner lines denote one standard error.

References

- Gašić, Milica, and Steve Young. "Gaussian processes for pomdp-based dialogue manager optimization." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.1 (2014): 28-40.

- ML Tutorial by Richard Turner: https://www.youtube.com/watch?v=92-98SYOdlY