1 Kalman Filter

$\begin{cases} X_{t+1} = FX_t + \epsilon_t & \epsilon_t \sim N(0,\Sigma_x) \\ Y_t = HX_t + \eta_t & \eta_t \sim N(0,\Sigma_y)\end{cases}$ $X_1 \sim N(\mu_p, \Sigma_p)$

Then if X_0 is Gaussian then $X_t|Y_{1:t} \sim N(\mu_t, \sigma_t)$:

$\mu_{t+1} = F\mu_t + K_{t+1}(y_{t+1} - HF\mu_t)$

$\Sigma_{t+1} = (I - K_{t+1}H)(F\Sigma_t F^T + \Sigma_x)$

$K_{t+1} = (F\Sigma_t F^T + \Sigma_x)H^T(H(F\Sigma_t F^T + \Sigma_x)H^T + \Sigma_y)^{-1}$

2 Gaussian Processes

$f \sim GP(\mu,k) \Rightarrow \forall\{x_1,\dots,x_n\}\ \forall n<\infty$

$[f(x_1)\dots f(x_n)] \sim N([\mu(x_1)\dots\mu(x_n)], K)$

where $K_{ij} = k(x_i,x_j)$

2.1 Gaussian Process Regression

$f\sim GP(\mu,k)$ then: $f|y_{1:n}, x_{1:n} \sim GP(\tilde\mu, \tilde k)$

$\tilde\mu(x) = \mu(x) + K_{A,x}^T(K_{AA}+\epsilon I_n)^{-1}(y_A - \mu_A)$

$\tilde k(x,x') = k(x,x') - K_{A,x}^T(K_A + \epsilon I_n)^{-1}K_{A,x'}$

Where: $K_{A,z} = [k(x_1,z)\dots k(x_n,z)]^T$

$[K_{AA}]_{ij} = k(x_i,x_j)$ and $\mu_A = [\mu(x_1\dots x_n)]^T$

2.2 Kernels

$k(x,y)$ is a kernel if it's symmetric semidefinite positive:
$\forall\{x_1,\dots,x_n\}$ then for the Gram Matrix $[K]_{ij} = k(x_i,x_j)$ holds $c^TKc \ge 0 \forall c$

Some Kernels: (h is the bandwidth hyperp.)

Gaussian (rbf): $k(x,y) = \exp(-\frac{\|x-y\|^2}{h^2})$

Exponential: $k(x,y) = \exp(-\frac{\|x-y\|}{h})$

Linear kernel: $k(x,y) = x^Ty$

2.3 Optimization of Kernel Parameters

Given a dataset A, a kernel function $k(x,y;\theta)$.

$y \sim N(0, K_y(\theta))$ where $K_y(\theta) = K_{AA}(\theta) + \sigma_n^2 I$

$\hat\theta = \arg\max_\theta \log p(y|X;\theta)$

In GP: $\hat\theta = \arg\min_\theta y^TK_y^{-1}(\theta)y + \log|K_y(\theta)|$

We can from here $\nabla\downarrow$:

$\nabla_\theta \log p(y|X;\theta) = \frac12 tr\left((\alpha\alpha^T - K^{-1})\frac{\partial K}{\partial\theta}\right), \alpha = K^{-1}y$

Or we could also be baysian about θ

2.4 Aproximation Techniques

3 Review of useful concepts and Introduction

3.1 Multivariate Gaussian

$f(x) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac12(x-\mu)^T\Sigma^{-1}(x-\mu)}$

Suppose we have a Gaussian random vector $X_V \sim N(\mu_V, \Sigma_{VV})$.
Suppose we take two disjoint subsets of V: $A = i_1,\dots,i_k$ and $B = j_1,\dots,j_m$.
Then, the conditional distribution:
$P(X_A|X_B = x_B) = N(\mu_{A|B}, \Sigma_{A|B})$ is Gaussian:

$\mu_{A|B} = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B)$

$\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$

3.2 Convex / Jensen's inequality

g(x) is convex $\Leftrightarrow x_1,x_2 \in \mathbb{R}, \lambda\in[0,1]: g''(x) > 0$
$g(\lambda x_1 + (1-\lambda)x_2) \le \lambda g(x_1) + (1-\lambda)g(x_2)$
$\varphi(E[X]) \le E[\varphi(X)]$

3.3 Review Probability

Probability space (Ω, F, P): Set of atomic events Ω. Set of all non-atomic events (σ-Algebra): $F \in 2^\Omega$. Probability measure: $P: F \to [0,1]$

Bayes' rule: $P(B|A) = P(A,B)/P(A) = P(A|B)P(B)/P(A)$, where $P(A) = \sum_b P(A|B)P(A)$

Union: $P(A\cup B) = P(A)+P(B)-P(A\cap B)$

Rules for joint distributions:

Sum rule (Marginalization):
$P(X_{1:i-1}, X_{i+1:n}) = \sum_{x_i} P(X_{1:i-1}, X_i = x_i, X_{i+1:n})$

Product rule (Chain rule):
$P(X_{1:n}) = P(x_1)P(X_2|X_1)\dots P(X_n|X_{1:n-1})$

Conditional Independence:
$X\perp Y|Z$ iff $P(X,Y|Z) = P(X|Z)P(Y|Z)$
If $P(Y|Z)>0 \Rightarrow P(X|Z,Y) = P(X|Z)$

Properties of Conditional Independence:
Symmetry: $X\perp Y \mid Z \Rightarrow Y\perp X\mid Z$
Decomposition: $X\perp(Y,W)\mid Z \Rightarrow X\perp Y\mid Z$
Contraction: $(X\perp Y\mid Z)\wedge(X\perp W\mid Y,Z) \Rightarrow X\perp Y,W\mid Z$
Weak union: $X\perp Y,W\mid Z \Rightarrow X\perp Y\mid Z,W$
Intersection: $(X\perp Y\mid W,Z)\wedge(X\perp W\mid Y,Z)\Rightarrow X\perp Y,W\mid Z$

4 Bayesian Networks

4.1 Basic concepts

A Bayesian network (G,P) consists of:
- A BN structure G (directed, acyclic graph)
- A set of conditional probability distributions
- (G,P) defines the joint distribution:
$P(X_1,\dots,X_n) = \prod_i P(X_i|Pa_{X_i})$

BNs with 3 nodes:

4.2 Active trails and d-separation

An undirected path in a BN structure G is called active trail for observed variables $O \in X_1,\dots,X_n$ of for every consecutive triple of variables X, Y, Z on the path:
- **indirect causal effect**:
$X\to Y\to Z$ and Y unobserved
- **indirect evidential effect**:
$X\leftarrow Y\leftarrow Z$ and Y unobserved
- **common cause**:
$X\leftarrow Y\to Z$ and Y unobserved.
- **common effect**:
$X\to Y\leftarrow Z$ and Y or any of Y's descendants is observed.

Any variables X_i and X_j for which there is no active trail for observations O are called d-separated by O.

Theorem: $d-sep(X_i;X_j|O)) \quad \Rightarrow X\perp Y|Z$

Converse does not hold in general!

5 Exact inference (tree-structured BN)

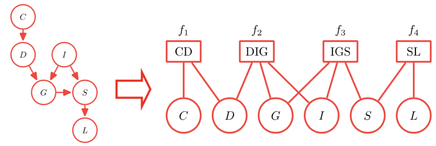
5.1 Variable elimination

- Given a BN and query $P(X|E=e)$
- Choose an ordering of $X_1,\dots,X_n$ **Eliminate variables from the outside in!**
- Set up initial factors: $f_i = P(X_i|Pa_i)$
- For $i=1:n, X_i\notin X,E$
  - Collect and multiply all factors f that include X_i
  - Generate new factor by marginalizing out X_i: $g_{X_i} = \sum_{x_i}\prod_j f_j$
  - Add g to set of factors
- Renormalize $P(x,e)$ to get $P(x|e)$

Variable elimination for polytrees:
- Pick a root, (avoiding X and E)
- Orient edges towards root
- Eliminate variables according to topological order

5.2 Avoiding recomputation: factor graphs

FG for a BN is a bipartite graph consisting of variables (circles) and factors (rectangles). **It is not a unique representation.**



5.2.1 Sum-product/Belief Propagation (BP) Algorithm:

- Initialize all messages as uniform distribution
- Until converged to:
  - Pick a root in the factor graph and reorient the edges towards this root.
  - Update messages according to this ordering. Do passes from leaves to root and from root to leaves.
- If a leaf node is a variable node: $\mu_{x\to f}(x) = 1$ -
If a leaf node is a factor node: $\mu_{f\to x}(x) = f(x)$ -
Messages from node v to factor u:
$\mu_{v\to u}(x_v) = \prod_{u'\in N(v)\setminus\{u\}}\mu_{u'\to v}(x_v)$
- Messages from factor u to node v:
$\mu_{u\to v}(x_v) = \sum_{x_u\sim x_v} f_u(x_u)\prod_{v'\in N(u)\setminus\{v\}}\mu_{v'\to u}(x'_v)$
  - Break once all messages change by $\le\epsilon$

Hope: after convergence, we have:

$P(X_v = x_v) = \frac1Z\prod_{u\in N(v)}\mu_{u\to v}(x_v)$

$P(\overrightarrow{X_u} = \overrightarrow{x_u}) = \frac1Z f_u(\overrightarrow{x_u})\prod_{v\in N(u)}\mu_{v\to u}(x_v)$

If we have a polytree Bayesian network:
- Choose one node as root
- Send messages from leaves to root and from root to leaves

6 Approximate inference (loopy networks)

With loopy graphs, BP is often **overconfident/oscillates**.

6.1 Variable elimination for MPE (most probable explanation):

- Given BN and evidence E=e
- Choose an ordering of $X_1,\dots x_n$
- Set up initial factors $f_i = P(X_i|Pa_i)$
- For $i=1:n, X_i\notin E$:
  - Collect and multiply all factors f_j that include X_i
  - Generate new factor by maximizing out X_i: $g_i = \max_{w=x_i}\prod_j f_j$
  - Add g to set of factors
- For $i = n-1:1, X_i\notin E$: $\hat x_i = \underset{x_i}{\arg\max}\, g_i(x_i, \hat x_{i+1:n})$

Retrieving MAP from Max-Product (MAP = MPE for a subset of RVs):
- Define max-marginals:
$P_{max}(X_v = x_v) := \max_{x\sim x_v} P(x)$
- For tree factor graphs, max-product computes max-marginals:
$P_{max}(X_v = x_v) \propto \prod_{u\in N(v)}\mu_{u\to v}(x_v)$
- Can retrieve MAP solution from these (must be careful when ties need to be broken).

6.2 Sampling based inference: compute marginals as expectations

Hoeffding's inequality: Suppose f is bounded in $[0,C]$. Then:
$P(|E_P[f(X)] - \frac1N\sum_{i=1}^N f(x_i)| \ge\epsilon) \le 2exp\left(\frac{-2N\epsilon^2}{C^2}\right)$

Monte Carlo Sampling from a BN:
- Sort variables in topological ordering X_1, X_n
- For $i=1$ to n, sample:
$x_i \sim P(X_1 = x_1,\dots,X_{i-1} = x_{i-1})$

Rejection Sampling:
- Collect samples over all variables:
$\hat P(X_A = x + A|X_B = x_B) = \frac{Count(x_A,x_B)}{Count(x_B)}$
- Throw away samples that disagree with x_B
- Count fraction of x_a on remaining samples

6.2.1 Directly sampling from the posterior: MCMC

Markov Chain:: A (stationary) MC is a sequence of RVs $X_1,\dots,X_N$, with prior $P(X_1)$ and transition probabilities $P(X_{t+1}|X_t)$ independent of t.

Markov assumpt.: $X_{1:t-1}\perp X_{t+1:T}|X_t, \forall t>1$

Stationarity assumption:
$P(X_{t+1} = x|X_t = x') = P(X_t = x|X_{t-1} = x'), \forall t>1$

If ergodic (= there exists a finite t such that every state can be reached in exactly t steps), then: it has a unique and positive stationary distribution $\pi(X)>0$, such for all x:
$\lim_{N\to\infty} P(X_N = x) = \pi(x)$ and $\pi(X)\perp P(X_1)$.

If MC satisfies the **detailed balance equation** (for unnormalized distribution Q, for all x,x':

$Q(x)P(x'|x) = Q(x')P(x|x')$), then the MC has stationarity distribution $\pi(X) = 1/Z Q(X)$.

Designing Markov Chains:

- Proposal distribution $R(X'|X)$: given $X_t = x$, sample "proposal" $x' \sim R(X'|X = x)$
- Acceptance distribution
  - Suppose $X_t = x$
  - With probability $\alpha = min\left\{1, \frac{Q(x')R(x|x')}{Q(x)R(x'|x)}\right\}$

set: $X_{t+1} = x'$
  - With probability $1 - \alpha$, set $X_{t+1} = x$

MCMC for graphical models: Gibbs sampling (Random Vs Practical variant):

- Start with initial assignment x to all variables
- Fix observed variables $X_B = x_B$
- For $t = 1$ to ∞, do:
  - Pick a variable i uniformly at random from $\{1, ..., n\} \setminus B$ / Set ordering, and then, for each X_i (except those in B)
  - Set v_i = values of all x except x_i
  - Sample x_i from $P(X_i|v_i)$

7 Dynamical models (include time)

7.1 Examples with one variable per time step

$X_1, ..., X_T$ (unobserved) hidden states

$Y_1, ..., Y_T$ (noisy) observations

HMMs (polytrees: can use belief propagation): X_i categorical, Y_i categorical (or arbitrary)

Kalman filters: X_i, Y_i Gaussian distributions

- $P(X_1)$: prior belief about location at time i
- $P(X_{t+1}|X_t)$: **'Motion model'** (how do I expect my target to move in the environment?): $X_{t+1} = FX + \epsilon_t$ where $\epsilon_t \sim N(0, \Sigma_x)$
- $P(Y_t|X_t)$: **'Sensor model'** (what do I observe if target is at location X_t?) $Y_t = HX_t + \eta_t$ where $\eta_t \sim N(0, \Sigma_y)$

7.2 Inference tasks

Filtering: $P(X_t|y_{1,...,t})$ Is it raining today?

Prediction: $P(X_{t+\tau}|Y_{1:t})$ Rain 5 days from now? Example for one step: $P(X_{t+1}|Y_{1:t}) = \sum_x P(X_{t+1}, X_t = x_t|Y_{1:t}) = \sum_x P(X_{t+1}|X_t = x_t)P(X_t|Y_{1:t})$ (with KFs, you need **integrals**!)

Smoothing: $P(X_\tau|y_{1:t})$ with $\tau < t$ Did it rain last week? [Can use sum-product (aka forward-backward).]

MPE: $\underset{x_{1:T}}{argmax} P(x_{1:T}|y_{1:T})$ Can use max product (aka Viterbi algorithm).

Bayesian filtering: Start with $P(X_1)$:

At time t, assume we have $P(X_t|y_{1:t-1})$

Conditioning: $P(X_t|y_{1:t}) = \frac{P(X_t|y_{1:t-1})P(y_t|X_t)}{\sum_{X_t} P(X_t|y_{1:t-1})P(y_t|X_t)}$

Prediction ($O(n^2)$ *vs* $O(n)$ in conditioning):

$P(X_{t+1}|y_{1:t}) = \sum_x P(X_{t+1}|X_t)P(X_t|y_{1:t})$

Since HMM is a polytree, smoothing/MPE can be computed by VE/BP. Kalman filtering: Bayesian filtering for continuous problems. RV corrupted by Gaussian distributions with zero mean. **Bayesian filtering is basically the same, except that sums turn to integrals. General Kalman update**

- Transition model: $P(x_{t+1}|x_t) = N(x_{t+1}; Fx_t, \Sigma_x)$
- Sensor model: $P(y_t|x_t) = N(y_t; Hx_t, \Sigma_y)$
- Kalman update:

$\mu_{t+1} = F\mu_t + K_{t+1}(y_{t+1} - HF\mu_t)$

$\Sigma_{t+1} = (I - K_{t+1})(F\Sigma_t F^T + \Sigma_x)$

- Kalman gain: $K_{t+1} = (F\Sigma_t F^T + \Sigma_x)H^T(H(F\Sigma_t F^T + \Sigma_x)H^T + \Sigma_y)^{-1}$

7.3 Examples with > 1 variable per time step

Dynamic Bayesian Networks: a BN at every time step

These models typically have many loops. Exact inference is usually intractable.

7.4 Approx. infer. for filtering (DBNs and non-linear Kalman filters): Particle filtering

Suppose: $P(X_t|y_{1:t}) \approx \frac{1}{N}\sum_{i=1}^N \delta_{x_{i,t}}$, where δ is the indicator function. **Prediction**: Propagate each particle: $x_i' \sim P(X_{t+1}|x_{i,t})$

Conditioning:

- weight particles $w_i = \frac{1}{Z}P(y_{t+1}|x_i')$
- resample N particles $x_{i,t+1} \sim \frac{1}{Z}\sum_{i=1}^N w_i\delta_{x_i'}$

Conclusion we came to: $Z = \sum_{i=1}^N w_i\delta_{x_i}$

8 Probabilistic Planning

8.1 Markov Decision Processes

An MDP is specified by a quintuple: $(X, A, r, P(x'|x,a), \gamma)$, where X are states, A are actions, $r(x,a)$ is a reward function and transition probabilities:

$P(x'|x,a) = \text{Prob(Next state} = x'|\text{Action } a)$

Objective: find a stationary policy $\pi : S \to A$ that maximizes the sum of cumulative rewards.

Value of a state given a policy : sum of cumulative rewards, given that the initial state is this state $\to$ **Bellman equation:**

$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^\infty \gamma^t r(s_t, \pi(s_t), s_{t+1})|s_0 = s\right]$

$= \sum_{s' \in S} P(s'|s,\pi(s))\left[r(s,\pi(s),s') + \gamma V^\pi(s')\right]$

$= r(s,\pi(s)) + \gamma\sum_{s' \in S} P(s'|s,\pi(s))V^\pi(s')$

Theorem (Bellman): a policy is optimal iff it is greedy w.r.t. its induced value function!

$V^*(x) = max_a[r(x,a) + \gamma\sum_{x'} P(x'|x,a)V^*(x')]$

Bellman equation mais geral:

$V^*(x) = max_a[\sum_{x'} P(x'|x,a)(r(a,x,x') + \gamma V^*(x'))]$

Optimal policy:

$\pi^*(s) = \underset{a \in A}{argmax}[r(x,a) + \gamma\sum_{x'} P(x'|x,a)V^*(x')]$

8.2 Policy iteration (Cost $O(S^3 + SA\Delta)$)

Start with an arbitrary (e.g. random) policy π. Until converged, do:

- Compute value function $V^\pi(x)$
- Compute greedy policy π_G w.r.t. V^π
- Set $\pi \leftarrow \pi_G$

Guaranteed to monotonically improve and to converge to an **optimal** policy π^* in $O(n^2 m/(1-\gamma))$ iterations (converges in polynomial number of iterations)!

8.3 Value iteration (Cost $O(SA\Delta)$)

Initialize $V_0(x) = max_a r(x,a)$

For $t = 1$ to ∞:

- For each (x,a), let:

$Q_t(x,a) = r(x,a) + \gamma\sum_{x'} P(x'|x,a)V_{t-1}(x')$

- For each x, let $V_t(x) = \underset{a}{max} Q_t(x,a)$
- Break if $\|V_t - V_{t-1}\|_\infty = \underset{x}{max}|V_t(x) - V_{t-1}(x)| \leq \epsilon$

Then choose greedy policy w.r.t V_t.

Guaranteed to converge to ϵ-optimal policy (finds approximate solution in polynomial number of iterations)!

8.4 POMDP = Belief-state MDP

States = beliefs over states for original POMDP

$B = \Delta(1, ..., n) = \{b : 1, ..., n \to [0,1], \sum_x b(x) = 1\}$

Actions: same as original MDP

Transition model:

- Stochastic observation:

$P(Y_t|b_t) = \sum_{x=1}^n P(Y_t|X_t = x)b_t(x)$

- State update (Bayesian filtering!), given b_t, y_t, a_t: $b_{t+1}(x') = \frac{1}{Z}\sum_x b_t(x)P(y_t|x)P(X_{t+1} = x'|X_t = x, a_t)$

Reward function: $r(b_t, a_t) = \sum_x b_t(x)r(x, a_t)$

8.5 Example of approx. solution to POMDPs: Policy gradients

- Assume parameterized policy: $\pi(b) = \pi(b;\theta)$
- For each parameter θ the policy induces a Markov chain
- Can compute expected reward $J(\theta)$ by sampling.
- Find optimal parameters through search (gradient ascent): $\theta^* = \underset{\theta}{argmax}\ J(\theta)$

9 Learning models from training data

9.1 Learning from i.i.d data

Algorithm for Bayes Net MLE:

Given BN of structure G and dataset D of complete observations

For each X_i estimate: $\hat{\theta}_{X_i|Pa_i} = \frac{Count(X_i, Pa_i)}{Count(Pa_i)}$

Pseudo-counts for lime and cherry flavor: $\theta_{F=c} \frac{Count(F=c) + \alpha_c}{N + \alpha_c + \alpha_l}$

9.1.1 Score based structure learning

Define scoring function $S(G;D)$ and search over BN structure G: $G^* = \underset{G}{argmax} S(G;D)$

Examples of scores:

MLE Score:

$log P(D|\theta_G, G) = N\sum_{i=1}^n \hat{I}(X_i; Pa_i) + const.$

Where mutual information ($I(X_i, X_j) \geq 0$) is:

$I(X_i, X_j) = \sum_{x_i, x_j} P(x_i, x_j) log\frac{P(x_i, x_j)}{P(x_i)P(x_j)}$

Empirical mutual information:

$\hat{P}(x_i, x_j) = \frac{Count(x_i, x_j)}{N}$

$\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) log\frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$

Regularizing a Bayes Net:

$S_{BIC}(G) = \sum_{i=1}^n \hat{I}(X_i; Pa_i) - \frac{logN}{2N}|G|$

where G is the number of parameters, n the number of variables and N the number of training examples.

Chow-Liu algorithm:

- For each pair X_i, X_j of variables, compute: $\hat{P}(x_i, x_j) = \frac{Count(x_i, x_j)}{N}$
- Compute mutual information
- Define complete graph with weight of edge (X_i, X_j) given by the mutual information
- Find max spanning tree $\to$ undirected tree
- Pick any variable as root and orient the edges away using breadth-first search.

10 Reinforcement Learning

10.1 Model-based RL

10.1.1 ϵ greedy

With probability ϵ, pick random action. With prob $(1-\epsilon)$, pick best action. If sequence ϵ satisfies Robbins Monro criteria $\to$ convergence to optimal policy with prob 1.

10.1.2 R_{max} algorithm

Input: starting x_0, discount factor γ.

Initially: add fairy tale state x^* to MDP

- Set $r(x,a) = R_{max}$ for all states x and actions a
- Set $P(x^*|x,a) = 1$ for all states x and actions a
- Choose the optimal policy for r and P

Repeat: 1. Execute policy π and, for each visited state/action pair, update $r(x,a)$

2. Estimate transition probabilities $P(x'|x,a)$

3. If observed 'enough' transitions/rewards, recompute policy π, according to current model P and r.

Enough"? See Hoeffding's inequality. To reduce error ϵ, need more samples N.

Theorem: With probability $1-\delta$, R_{max} will reach an ϵ-optimal policy in a number of steps that is polynomial in $|X|, |A|, T, 1/\epsilon$ and $log(1/\delta)$.

Memory $O(|X^2||A|)$.

10.2 Model-free RL: estimate V*(x) directly

10.2.1 Q-learning

$Q(x,a) \leftarrow (1-\alpha_t)Q(x,a)+\alpha_t(r+\gamma \max_{a'} Q(x',a'))$

Theorem: If learning rate α_t satisfies: $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$ (Robbins-Monro), and actions are chosen at random, then Q learning converges to optimal Q^* with probability 1.

Optimistic Q learning:

Initialize: $Q(x,a) = \frac{R_{max}}{1-\gamma} \prod_{t=1}^{T_{init}} (1-\alpha_t)^{-1}$

Same convergence time as with R_{max}. Memory $O(|X||A|)$. Comp: $O(|A|)$.

Parametric Q-function approximation: $Q(x,a;\theta) = \theta^T \phi(x,a)$ to scale to large state spaces. (You can use Deep NN here!)

SGD for ANNs: initialize weights. For t = 1,2..., pick a data point (x,y) uniformly at random. Take step in negative gradient direction. (In practise, mini-batches).

Deep Q Networks: use CNN to approx Q function. $L(\theta) = \sum_{(x,a,r,x')\in D}(r+\gamma \underset{a'}{max} Q(x',a';\theta^{old})-Q(x,a;\theta))^2$ **Double DQN:** current network for evaluating argmax (too optimistic, and you remove θ^{old} and put θ).

10.3 Gaussian processes

A GP is an (infinite) set of random variables (RV), indexed by some set X, i.e., for each x in X, there is a RV Y_x where there exists functions $\mu : X \rightarrow \mathbb{R}$ and $K : X \times X \rightarrow \mathbb{R}$ such that for all: $A \in X, \quad A = x_1, ...x_k$, it holds that $Y_A = [Y_{x_1}, ..., Y_{x_k}] \sim N(\mu_a, \Sigma_{AA})$, where: $\Sigma_{AA} =$ matrix with all combinations of $K(x_i, x_j)$.

K is called kernel (covariance) function (must be symmetric and pd) and μ is called mean function. **Making prediction with GPs:** Suppose $P(f) = GP(f;\mu,K)$ and we observe $y_i = f(\overrightarrow{x_i}) + \epsilon_i$, $A = \{\overrightarrow{x_1} : \overrightarrow{x_k}\}$ $P(f(x)|\overrightarrow{x_1} : \overrightarrow{x_k}, y_{1:k}) = GP(f;\mu',K')$. In particular, $P(f(x)|\overrightarrow{x_1} : \overrightarrow{x_k}, y_{1:k}) = N()f(x); \mu_{x|A}, \sigma^2_{x|a}$, where $\mu_{x|a} = \mu(\overrightarrow{x})+\Sigma_{x,A}(\Sigma_{AA}+\sigma^2 I)^{-1}\Sigma_{x,A}^T(\overrightarrow{y_A}-\mu_A)$ and $\sigma^2_{x|a} = K(\overrightarrow{x},\overrightarrow{x})-\Sigma_{x,A}(\Sigma_{AA}+\sigma^2 I)^{-1}\Sigma_{x,A}^T$. **Closed form formulas for prediction!**