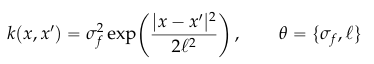
# 3.

## a)

1) 

2) ??? Unknown?

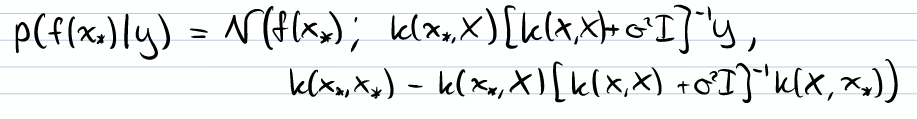
3) length scale

4) marginal likelihood

5) 

6) Complexity

7) Fitting data

8, 9, 10) 

11) O(n^3) (to calculate [K + σ I]-1)

12) O(n^2) (to do the multiplications)

## b) i.

2nd clearly has much larger length scale and noise variance - the first is much less and more appropriate (within range of data)

## b) ii.

These hyperparameters are chosen by doing numerical optimization of the marginal likelihood (analytical is intractable). This optimization can easily fall into the local optimum that is the second one - as it has a decent marginal likelihood.

## b) iii.

* if we integrate hyper-parameters we would get an exact posterior
* Analytically if possible, otherwise monte carlo would be applicable here.

## c) is this assessed?

# 4.

## a)

i. No as f is dependent on a

ii. Yes no dependencies involved

iii. No (idk why it’s no)

iv. Yes d and j are both dependent on g but not h

v. No (idk why it's no)

vi. No d and j share a parent node

vii. No (idk why it’s no as i is dependent on h and c is dependent on f but it’s not both)

## b) i.

* Train GP on initial dataset (from objective function)
* Find argmax of acquisition function (applied to GP)
* Compute new value of objective function at this argmax
* Add new value to dataset
* Repeat

## b) ii.

To choose where to evaluate the objective function next. Must balance the tradeoff between exploration and exploitation.

## b) iii.

The main challenge is optimizing the acquisition function, especially since gradients are calculated approximately (e.g. finite difference) - this was the case in the coursework, so only small steps must be taken.

This means a lot of evaluations of the GP must occur, which for 10000 values is very expensive.

## b) iv.

Location is scored based on the probability is it better than the current best: p(f(x) < f(xbest)). This can be calculated directly from the GP as Φ(γ(x)).

This means we optimise for exploitation (only), so if there’s a good chance we’ll find a better value in a given location, we’ll try there. Makes no attempt at exploration, as if we have a high std in a location, this will likely give a very low PI.

To fix this lack of exploration, we can add a slack variable, the score becomes based on the probability is it better than the current best, lose a slack amount. I.e. small improvements are not rewarded, big improvements are.

Stated as: p(f(x) < f(xbest) - ζ), calculated as: 

## c) ii.

### A)

We just need to ensure the mean of kq(x) is the same as p(x), since if the std is larger, the rest of the distribution will be larger.

If both mean and x are 0 for a MVN, the exponential term is just 1. So we only need to consider the normalisers.

kq(0) = p(0)

k (2π)-0.5n |Σq|-0.5 = (2π)-0.5n |Σp|-0.5

k |Σq|-0.5 = |Σp|-0.5

k |I σ2q|-0.5 = |I σ2p|-0.5

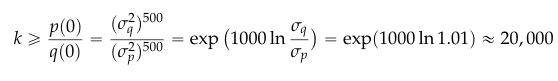
k σq-n = σp-n

k (1.01σq)-n = σp-n

k (1.01)-n = 1

k = 1.01n

And if n = 1000, we get k = 20960



### B)

Ratio of volumes

1/k = 0.0000477