Lecture 20: Gaussian Processes

What we did so far:

observe data: $[x_1, y_1], [x_2, y_2], \ldots$

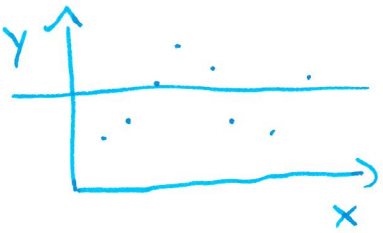
assume model, e.g. $y_i = \theta^T x_i + \epsilon_i$

estimate parameters: $p(\theta|D) \propto p(D|\theta) \cdot p(\theta)$

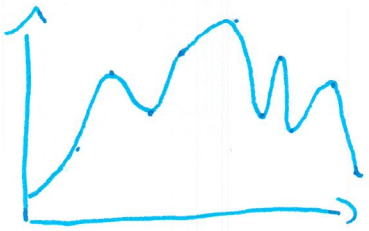
make predictions for new data inputs

$p(y_* | x_*, x, y)$

⇒ A problem with this approach is coming up with the right model ⇒ might underfit or overfit



underfit



overfit

⇒ It would be great if instead of estimating θ, we could estimate the model.

⇒ We need a distribution over functions $f(x)$:

$$y_i = f(x_i) + \epsilon_i$$

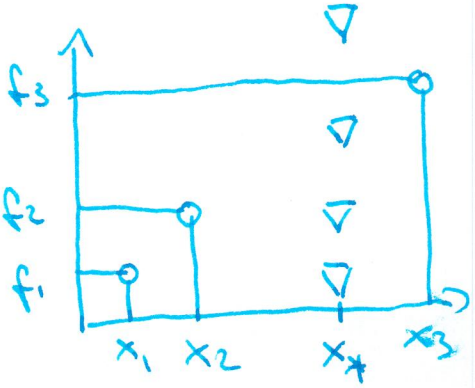
$$p(y_* \mid x_*, X, y) = \int p(y_* \mid f, x_*) \cdot p(f \mid X, y)\, df$$

⇒ This looks rather difficult, but it is exactly what Gaussian Processes enable us to do!

Here is the trick: think about functions as lookup tables.

⇒ We only need to be able to define a distribution over the function's values at a finite, but arbitrary set of points.

Let us look at an example:



which ▽ would you choose for f_*?

⇒ Intuitively we want our function to be smooth.

⇒ points that are closer in x_i should be closer together in f_i.

⇒ x_1 gives us information about x_2, and a bit less information about x_3.

In the Gaussian Process setting we model $p(f_1, f_2, f_3)$ as jointly Gaussian:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \sim N\left(\underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\mu}, \underbrace{\begin{bmatrix} 1 & 0.7 & 0.2 \\ 0.7 & 1 & 0.6 \\ 0.2 & 0.6 & 1 \end{bmatrix}}_{K} \right)$$

Note: These numbers are made up, it is about understanding the concept.

⇒ Covariance captures similarity

⇒ $\text{cov}(x_1, x_2) > \text{cov}(x_1, x_3)$

How can we construct the covariance matrix K?

⇒ We need some measure of similarity, and K needs to be positive definite.

⇒ K can be computed using standard covariance functions / kernels.

⇒ One popular choice is the squared exponential:

$$K_{ij} = \tau^2 \cdot e^{\frac{-\|x_i - x_j\|^2}{2\lambda^2}}$$

↑ not always there

⇒ $K_{ij} = 1$ if $x_i = x_j$

λ is a parameter that regulates the smoothness of our function / GP.

τ regulates the height.

⇒ Once again there is no free lunch. We have to decide for a kernel and tune the parameters.

⇒ So far all x_i are given, and we can use our covariance function to construct K.

⇒ K is also called the Gram matrix.

How do we set a prediction f_* for a given new point x_*?

Assumption: The new unseen data point comes from the same distribution as the training data:

$$f_* \sim N(0, K_{**})$$

$$K_{**} = \tau^2 e^{\frac{-\|x_* - x_*\|^2}{2\lambda^2}} = 1$$

⇒ If we define our problem like this, then the known f_i and f_* are independent, but we want smoothness, so they should be correlated.

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} & \begin{bmatrix} K_{1*} \\ K_{2*} \\ K_{3*} \end{bmatrix} \\ \begin{bmatrix} K_{*1}, & K_{*2}, & K_{*3} \end{bmatrix} & \begin{bmatrix} K_{**} \end{bmatrix} \end{bmatrix} \right)$$

K, K_*, K_{**}, K_*^T

⇒ What we have now is a joint distribution

$$p(f_*, f_1, f_2, f_3 \mid X_*, X).$$

⇒ What we want is the conditional

$$p(f_* \mid f, X_*, X).$$

⇒ We remember HW5 and that the conditionals of a multivariate Gaussian are also Gaussian.

$$\Rightarrow p(f_* \mid f, X_*, X) \sim \mathcal{N}(\mu_*, \Sigma_*)$$

$$\mu_* = \mu(X_*) + K_*^T K^{-1} (f - \mu(X))$$

$$\Sigma_* = K_{**} - K_*^T K^{-1} K_*$$

It is common to just set the mean of the joint distribution to zero, because the GP is flexible enough to model μ* arbitrary well. If you have a special case and really want to model the mean using a parametric function, please see

Kevin P. Murphy, section 15.2.6.

So far we have looked at the noiseless case. What if we consider our noisy

$$y_i = f_i + \epsilon_i = f(x_i) + \epsilon_i \; ?$$

⇒ The noise is independently added to each observation

⇒ Noise only influences the diagonal of the covariance matrix

$$K_{ij} = k(x_i, x_j) + \sigma_\epsilon^2 \delta_{ij}$$

↳ Kernel ↳ Kronecker delta

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Assuming the conventional zero mean for the joint distribution, we get the following conditionals:

$$p(f_*^* | X_*, X, Y) = N(\mu_*, \Sigma_*)$$

$$\mu_* = K_*^T K_y^{-1} \cdot Y$$

$$\Sigma_* = K_{**} - K_*^T K_y^{-1} K_*$$

$\Rightarrow$ K_y is K plus the noise values for the diagonal.

A note about the matrix inverse:

Inverting a matrix can be tricky / numerically unstable. A more stable possibility is to use a cholesky decomposition of the matrix K.

Cholesky decomposition:

$K = L L^T$ L is a lower triangular matrix with real an positive diagonal entries.

$$\mu_* = K_*^T \cdot K_Y^{-1} \cdot Y$$
$$= K_*^T \cdot (L \cdot L^T)^{-1} Y$$
$$= K_*^T \cdot \underbrace{L^{-T} \cdot \underbrace{L^{-1} \cdot Y}_{m}}_{\alpha}$$

$m = L^{-1} Y \iff Lm = Y \Rightarrow$ solve linear system of equations to get m

$\alpha = L^{-T} \cdot m \iff L^T \cdot \alpha = m$

$\Rightarrow \mu_* = K_*^T \cdot \alpha$

Learning the Kernel parameter

marginal likelihood:

$$p(y|X,\theta) = \int p(y|f,X,\theta) \cdot p(f|X,\theta)\, df$$

$$p(f|X,\theta) = N(0,K)$$

$$p(y|f,\theta) = \Pi_i\, N(f_i, \sigma_y^2)$$

$$\log(p(y|X,\theta)) = -\frac{1}{2} y K_y^{-1} y - \frac{1}{2}\log|K_y| - \frac{N}{2}\log(2\pi)$$

⇒ We can use optimization methods and do maximum likelihood, or specify priors for the parameters θ and so for the posterior.

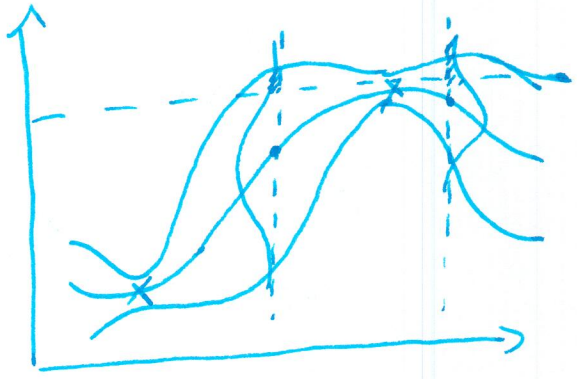
Active learning using GP

- Scenario: We want to find the maximum of a function, but querying points is expensive.

  ⇒ Need to balance exploration and exploitation

  Example: drilling for oil
  one armed bandit in a casino
  winning a chess game

Probability of improvement:



PI looks at the area of the sample Gaussian that is above the f_{max} line.

$$PI(x) = p(f(x) \geq \mu^+ + \xi)$$

$\Rightarrow$ Offset ξ is necessary to deal with the case that we are looking at μ^+ again.

$$PI(x) = \Phi\left(\frac{\mu(x) - \mu^+ - \xi}{\sigma(x)}\right)$$ Φ: CDF of the Gaussian.

PI is not often used in practice, but if you know the best you can get, it can be very powerful.

Expected Improvement

This criterion maximizes expected utility/ minimizes expected cost.

Example:

$p(x = \text{healthy} \mid \text{data}) = 0.9$

$p(x = \text{cancer} \mid \text{data}) = 0.1$

	no treatment	treatment	
healthy	0	-30	$u(x, a)$
cancer	-100	-20	

$$EU(a) = \sum_x u(x, a) \cdot p(x \mid data)$$

$$EU(a = treatment) = u(healthy, treatment) \cdot p(healthy \mid data) + u(cancer, treatment) \cdot p(cancer \mid data)$$

⇒ In this example the state is healthy or cancer, now the state is a function that we sample from our GP.

$$\underset{x}{\operatorname{argmin}} \int \| f_{n+1}(x) - f(x^*) \| \cdot p(f_{n+1} \mid D) \, df_{n+1}$$

f_{n+1}: from our GP; f: true function, maximum at x^*

⇒ find the point that minimizes the expected distance to the true maximum over all

possible functions from our GP.

But we don't know the true function!

⇒ Use the best value we observed so far instead:

$$x_{n+1} = \underset{x}{\operatorname{argmax}}\ \mathbb{E}\left[\max\{0, f_{n+1}(x) - f^{max}\} \mid D_n\right]$$

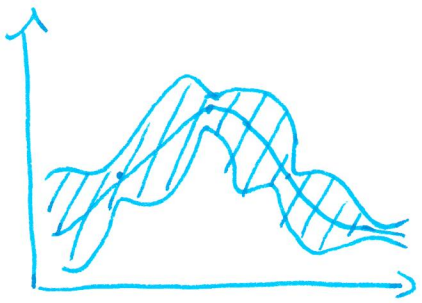
⇒ We want to do better than the f^{max} we have seen before.

It turns out that you can solve this expression analytically:

$$EI(x) = \begin{cases} (\mu(x) - \mu t - \xi)\,\underbrace{\Phi(z)}_{\text{CDF}} + \sigma(x)\,\underbrace{\varphi(z)}_{\text{PDF}} & \text{if } \sigma(x) > 0 \\ 0 \text{ if } \sigma(x) = 0 \end{cases}$$

$$z = \frac{\mu(x) - \mu t - \xi}{\sigma(x)}$$

Thompson sampling



If I sample a function from the GP, it has a high prob. of staying within the shaded area.

⇒ draw a function and choose the maximum of this function as the next point.

⇒ Good points create bottlenecks at good regions
⇒ exploitation.

⇒ Google analytics is using this.

Acknowledgement: This lecture is heavily based on Kevin P. Murphy, and on Nando de Freitas lectures in CPSC 540.
Nando is one of the best instructors I know and I highly recommend watching his Youtube videos.