Surrogate Optimization: GPR Hyperparameters

BIOE 498/598 PJ

Spring 2021

Gaussian Process Regression: nonparametric Bayesian optimization

- We assume the *inverse exponentiated squared Euclidean distance* kernel:

$$\Sigma(x, x') = \exp\{-\|x - x'\|^2\}.$$

- Given training data (X_n, y_n), predictions y at a new point x are

$$y(x) = \Sigma(x, X_n)\Sigma_n^{-1} y_n.$$

- The variance σ^2 at the points x can also be computed:

$$\sigma^2(x) = \Sigma(x, x) - \Sigma(x, X_n)\Sigma_n^{-1}\Sigma(x, X_n)^\top.$$

Gaussian Process Regression: nonparametric Bayesian optimization

- We assume the *inverse exponentiated squared Euclidean distance* kernel:

$$\Sigma(x, x') = \exp\{-\|x - x'\|^2\}.$$

- Given training data (X_n, y_n), predictions y at a new point x are

$$y(x) = \Sigma(x, X_n)\Sigma_n^{-1} y_n.$$

- The variance σ^2 at the points x can also be computed:

$$\sigma^2(x) = \Sigma(x, x) - \Sigma(x, X_n)\Sigma_n^{-1}\Sigma(x, X_n)^\top.$$

- GPR is *Bayesian*: the kernel (prior) is updated with data (X_n, y_n) to compute posterior estimates of (x, y).
- GPR is *nonparametric* since the "Kriging" equations for $y(x)$ and $\sigma^2(x)$ do not contain parameters.

Gaussian Process Regression: nonparametric Bayesian optimization

- We assume the *inverse exponentiated squared Euclidean distance* kernel:

$$\Sigma(x, x') = \exp\{-\|x - x'\|^2\}.$$

- Given training data (X_n, y_n), predictions y at a new point x are

$$y(x) = \Sigma(x, X_n)\Sigma_n^{-1} y_n.$$

- The variance σ^2 at the points x can also be computed:

$$\sigma^2(x) = \Sigma(x, x) - \Sigma(x, X_n)\Sigma_n^{-1}\Sigma(x, X_n)^\top.$$

- GPR is *Bayesian*: the kernel (prior) is updated with data (X_n, y_n) to compute posterior estimates of (x, y).
- GPR is *nonparametric* since the "Kriging" equations for $y(x)$ and $\sigma^2(x)$ do not contain parameters.
- In practice, we can use a few *hyperparameters* to improve the performance of GPR.

Scale

- GPR makes predictions by drawing from a multivariate normal distribution. This mean most predictions will lie in $[-2, 2]$.
- For our sinusoidal example, the response was in $[-1, 1]$, so we never noticed a problem. But not all problems have nice scaling.

Scale

- GPR makes predictions by drawing from a multivariate normal distribution. This mean most predictions will lie in $[-2, 2]$.
- For our sinusoidal example, the response was in $[-1, 1]$, so we never noticed a problem. But not all problems have nice scaling.
- Previously, the covariance matrix Σ was defined based on a correlation function
$$C(x, x') = \exp\{-\|x - x'\|^2\}.$$
- Let's scale the correlation function by a hyperparameter τ^2:
$$\Sigma = \tau^2 C(x, x') = \tau^2 \exp\{-\|x - x'\|^2\}$$

Scale

- GPR makes predictions by drawing from a multivariate normal distribution. This mean most predictions will lie in $[-2, 2]$.
- For our sinusoidal example, the response was in $[-1, 1]$, so we never noticed a problem. But not all problems have nice scaling.
- Previously, the covariance matrix Σ was defined based on a correlation function

$$C(x, x') = \exp\{-\|x - x'\|^2\}.$$

- Let's scale the correlation function by a hyperparameter τ^2:

$$\Sigma = \tau^2 C(x, x') = \tau^2 \exp\{-\|x - x'\|^2\}$$

- Where do we get τ^2? From the data! The maximum likelihood estimate is

$$\hat{\tau}^2 = \frac{y_n^\top C_n^{-1} y_n}{n}$$

- We'll let a software package handle these estimates for us.

Nugget

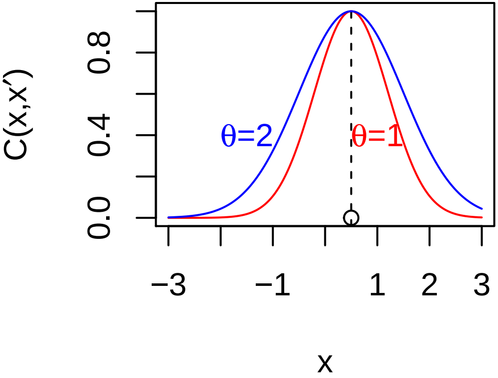
- Our correlation function $C(x, x')$ assumes that $y(x') \to y(x)$ as $x' \to x$.
- GPR always "connects the dots". If we measures $y(x)$, the GPR prediction at x will be the measured value.
- With real experiments, our measurements of y will be noisy, and we want GPR to smooth over this noise. Also, what if we made repeated measurements at y? What will the prediction be?
- **Solution:** Break the perfect correlation in C by injecting a small amount of white noise.
- **Method:** Add a "nugget" g to the diagonal of Σ_n:

$$\Sigma_n = \tau^2[C(X_n, X_n) + g\mathbb{I}].$$

Lengthscale

- GPR require the correlation function $C(x, x')$ to quantify how quickly the relationship between points decays.
- The lengthscale of decay can also be tuned by adding a hyperparameter θ:

$$C(x, x') = \exp\left\{-\frac{\|x - x'\|^2}{\theta}\right\}.$$



Putting it all together

- If we only want to include a scale τ^2, we can estimate its value using

$$\hat{\tau}^2 = \frac{y_n^\top C_n^{-1} y_n}{n}.$$

Putting it all together

- If we only want to include a scale τ^2, we can estimate its value using

$$\hat{\tau}^2 = \frac{y_n^\top C_n^{-1} y_n}{n}.$$

- Since the nugget g and the lengthscale θ alter C_n, all three of the parameters must be estimated simultaneously by optimization.
- Note that this optimization requires inverting Σ_n at each iteration. **Tuning a GPR model can be more expensive than training it!**

Putting it all together

- If we only want to include a scale τ^2, we can estimate its value using

$$\hat{\tau}^2 = \frac{y_n^\top C_n^{-1} y_n}{n}.$$

- Since the nugget g and the lengthscale θ alter C_n, all three of the parameters must be estimated simultaneously by optimization.
- Note that this optimization requires inverting Σ_n at each iteration. **Tuning a GPR model can be more expensive than training it!**
- Now is a good time to offload all the computation to a GPR library.

The `laGP` library

We will use four functions from the `laGP` library:

- `newGP`: trains a GPR model with initial data.
- `jmleGP`: tunes the model's hyperparameters jointly by maximum likelihood estimation.
- `predGP`: predicts the response at new inputs.
- `deleteGP`: deletes the model and releases memory when we're done.

laGP in practice: Initial training

Let's start with our previous training data (X_n, y_n).

```
Xn <- matrix(seq(-3,3,0.8), ncol=1)
yn <- sin(Xn[ ,1])
```

laGP in practice: Initial training

Let's start with our previous training data (X_n, y_n).

```
Xn <- matrix(seq(-3,3,0.8), ncol=1)
yn <- sin(Xn[ ,1])
```

Now we train a GPR model with laGP.

```
library(laGP)
gp <- newGP(Xn, yn, d=1, g=0.1*var(yn), dK=TRUE)
```

- d$=\theta$, which we initially set to 1.
- Our initial guess for the nugget g is 10% of the variance in the response.
- dK=TRUE makes derivatives of the kernel available so we can use MLE for hyperparameter tuning.

laGP in practice: Hyperparameter tuning

```
mle <- jmleGP(gp, drange=c(0,2), grange=c(0,var(yn)))
```

- `param=c("d","g")` asks `mleGP` to tune both the lengthscale and nugget. The scale τ^2 is tuned automatically.
- `drange` and `grange` are vectors of bounds for θ and g.
  - We want a range large enough to avoid hitting the bounds, but small enough to make the search efficient.
  - $\theta = 2$ is relatively large for our problem since $-3 \leq x \leq 3$.
  - The nugget g is rarely larger than the variance of the training responses.

`laGP` in practice: Hyperparameter tuning

```
mle <- jmleGP(gp, drange=c(0,2), grange=c(0,var(yn)))
```

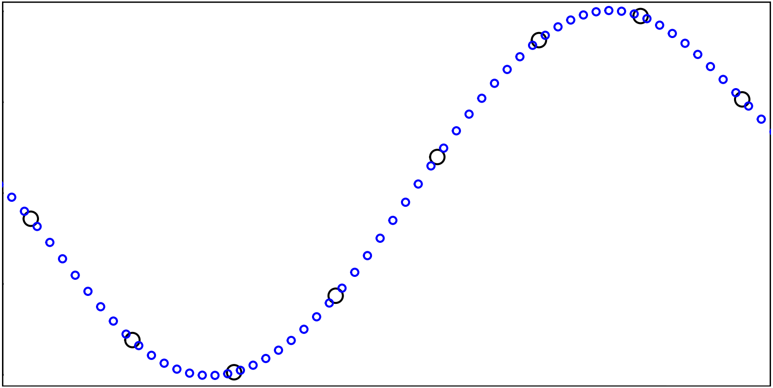
- `param=c("d","g")` asks `mleGP` to tune both the lengthscale and nugget. The scale τ^2 is tuned automatically.
- `drange` and `grange` are vectors of bounds for θ and g.
  - We want a range large enough to avoid hitting the bounds, but small enough to make the search efficient.
  - $\theta = 2$ is relatively large for our problem since $-3 \leq x \leq 3$.
  - The nugget g is rarely larger than the variance of the training responses.

```
mle
```

```
##   d           g tot.its dits gits
## 1 2 5.41316e-09    3817 3778   39
```

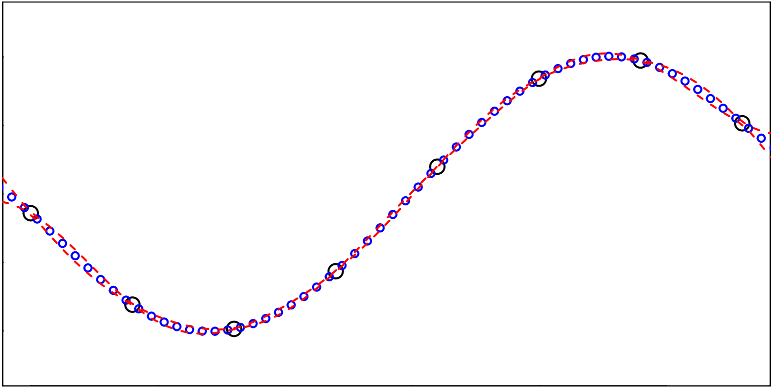
laGP in practice: Prediction

```
X <- matrix(seq(-3.25,3.15,0.1), ncol=1)
yp = predGP(gp, X)
par(mar=rep(0,4))
plot(Xn,yn)
points(X,yp$mean, col="blue", cex=0.5)
```



laGP in practice: Uncertainty

```
s2 <- diag(yp$Sigma)
par(mar=rep(0,4))
plot(Xn,yn, ylim=c(-1.3,1.3))
points(X,yp$mean, col="blue", cex=0.5)
lines(X, yp$mean + qnorm(0.05, 0, sqrt(s2)), lty=2, col=2)
lines(X, yp$mean + qnorm(0.95, 0, sqrt(s2)), lty=2, col=2)
```



`laGP` in practice: Cleanup

The `laGP` model is stored in a external C library, so R cannot delete it directly. We need to call `deleteGP` to avoid a memory leak.

```
deleteGP(gp)
```

Extra information: Anisotropy

- It's assumed that the scale τ^2 and nugget g are constant over the entire search space.
- The lengthscale θ may not be. In particular, θ could be different for each dimension.
- Imagine optimizing reaction yield based on time, temperature, and substrate. Small changes in temperature may have big effects (small θ), while the yield may be insensitive to changes in time (large θ).
- Dimensions with longer lengthscales require fewer data for prediction.

Extra information: Anisotropy

- It's assumed that the scale τ^2 and nugget g are constant over the entire search space.
- The lengthscale θ may not be. In particular, θ could be different for each dimension.
- Imagine optimizing reaction yield based on time, temperature, and substrate. Small changes in temperature may have big effects (small θ), while the yield may be insensitive to changes in time (large θ).
- Dimensions with longer lengthscales require fewer data for prediction.

- An *isotropic* model assumes parameters are fixed over all dimensions. An *anisotropic* model assumes parameters like θ vary by dimension. Anisotropic models are also called *separable*.
- For anisotropic models we estimate a vector of lengthscales, one for each dimension. The anisotropic correlation function is

$$C(x, x') = \exp\left\{-\sum_{k=1}^{m} \frac{(x_k - x'_k)^2}{\theta_k}\right\}.$$

Extra information: Anisotropy

- It's assumed that the scale τ^2 and nugget g are constant over the entire search space.
- The lengthscale θ may not be. In particular, θ could be different for each dimension.
- Imagine optimizing reaction yield based on time, temperature, and substrate. Small changes in temperature may have big effects (small θ), while the yield may be insensitive to changes in time (large θ).
- Dimensions with longer lengthscales require fewer data for prediction.

- An *isotropic* model assumes parameters are fixed over all dimensions. An *anisotropic* model assumes parameters like θ vary by dimension. Anisotropic models are also called *separable*.
- For anisotropic models we estimate a vector of lengthscales, one for each dimension. The anisotropic correlation function is

$$C(x, x') = \exp\left\{-\sum_{k=1}^{m} \frac{(x_k - x'_k)^2}{\theta_k}\right\}.$$

- `laGP` provides separate functions for anisotropic models: `newGPsep`, `jmleGPsep`, `predGPsep`, `deleteGPsep`.

Summary

- For best performance, GPR models must be tuned to find a scale τ^2, nugget g, and lengthscale θ that matches the training data.
- Tuning is computationally expensive. Often we tune following the initial training and then only update (`updateGP`) as subsequent data are collected.
- Anisotropic models can also increase performance, but we will focus on isotropic models in this course.

Summary

- For best performance, GPR models must be tuned to find a scale τ^2, nugget g, and lengthscale θ that matches the training data.
- Tuning is computationally expensive. Often we tune following the initial training and then only update (`updateGP`) as subsequent data are collected.
- Anisotropic models can also increase performance, but we will focus on isotropic models in this course.

- **Next time:** Given a GPR model, where should our next experiment be?