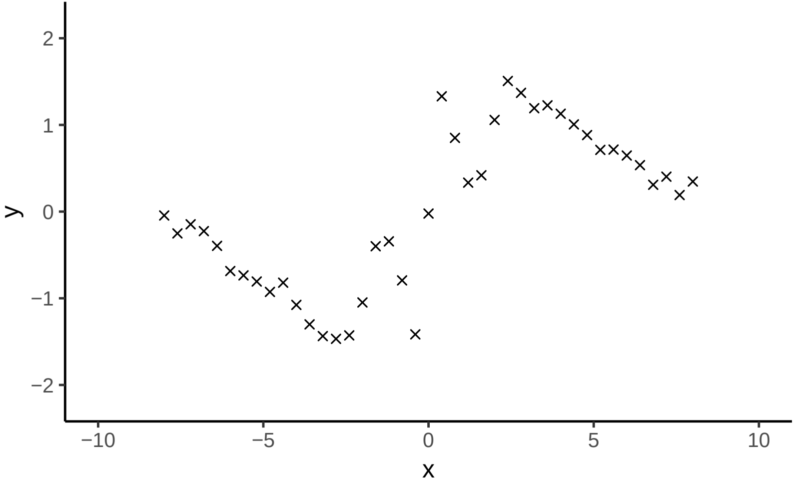
Gaussian Processes

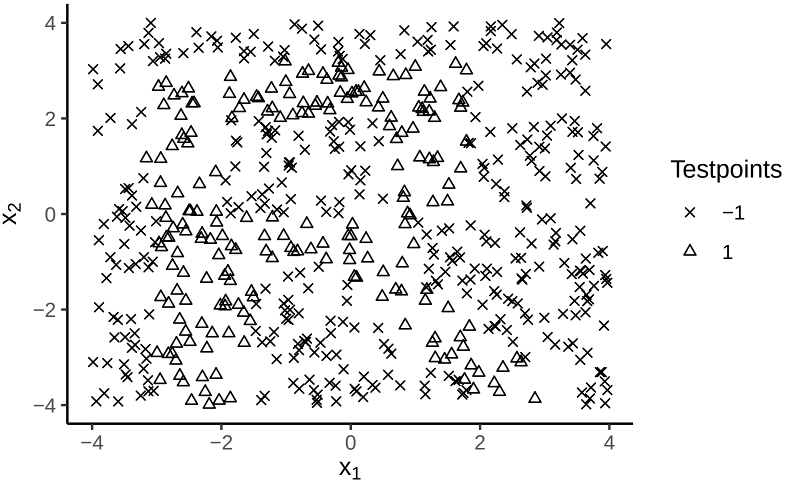
Budjan, Haas, Klumpp, Reitze

Datum: 22. Februar 2019

Regression und Klassifikation



Regression und Klassifikation



Einführung zu Gaußschen Prozessen

- Stochastischer Prozess: $f : \Omega \times X \to Z$, $(\omega, x) \mapsto f_x(\omega)$ wobei $f_x(\omega)$ messbar in ω für alle $x \in X$.

Einführung zu Gaußschen Prozessen

- Stochastischer Prozess: $f : \Omega \times X \to Z$, $(\omega, x) \mapsto f_x(\omega)$ wobei $f_x(\omega)$ messbar in ω für alle $x \in X$.

- Anschaulich: Stochastischer Prozess ist Zufallsvariable mit Werten f_x in Funktionenraum

Einführung zu Gaußschen Prozessen

- Stochastischer Prozess: $f : \Omega \times X \to Z$, $(\omega, x) \mapsto f_x(\omega)$ wobei $f_x(\omega)$ messbar in ω für alle $x \in X$.

- Anschaulich: Stochastischer Prozess ist Zufallsvariable mit Werten f_x in Funktionenraum

- Ein Gaußscher Prozess ist ein stochastischer Prozess $(f_x)_{x \in X}$ wobei für jede endliche Teilmenge $Y \subset X$, $(f_x)_{x \in Y}$ multivariat normal verteilt ist.

Einführung zu Gaußschen Prozessen

- Stochastischer Prozess: $f : \Omega \times X \to Z$, $(\omega, x) \mapsto f_x(\omega)$ wobei $f_x(\omega)$ messbar in ω für alle $x \in X$.

- Anschaulich: Stochastischer Prozess ist Zufallsvariable mit Werten f_x in Funktionenraum

- Ein Gaußscher Prozess ist ein stochastischer Prozess $(f_x)_{x \in X}$ wobei für jede endliche Teilmenge $Y \subset X$, $(f_x)_{x \in Y}$ multivariat normal verteilt ist.

- Wird charakterisiert durch $m(x) = \mathbb{E}(f(x))$ und $k(x, x') = \mathrm{Cov}(f(x), f(x'))$
  Im Folgenden: $X = \mathbb{R}^d$ und $m(x) = 0$
  Beispiel für k: $k(x, y) = \exp\left(-\frac{(x-y)^2}{2\ell}\right)$

Theorie zu Regression

Gemeinsame Verteilung für Datenpunkte $(X, f(X))$ und Testpunkte:

$$\begin{pmatrix} f(X) \\ f(X_*) \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{pmatrix}\right)$$

$K(X,X)_{i,j} = k(X_i, X_j)$
Bedingte Verteilung:

$$f(X_*)|f(X), X, X_* \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu = K(X,X_*)K(X,X)^{-1}f(X)$$

$$\Sigma = K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)$$

predict Algorithmus

Annahme: $y_i = f(x_i) + \varepsilon_i$, $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_n^2)$

Inputs: X (inputs), y (targets), σ_n^2 (noise),
K (covariance funtion), X_* (test input)

$$
\begin{array}{ll}
\# \, 1 & L = \mathrm{cholesky}(K(X,X) + \sigma_n^2 I) \\
\# \, 2 & \alpha = \mathrm{solve}(L^\top, \mathrm{solve}(L, y)) \\
\# \, 3 & \bar{f}(X_*) = K(X, X_*)^\top \cdot \alpha \\
\# \, 4 & v = \mathrm{solve}(L, K(X, X_*)) \\
\# \, 5 & \overline{\mathrm{V}}(\bar{f}(X_*)) = K(X_*, X_*) - v^\top v
\end{array}
$$

return: $\bar{f}(X_*)$, $\overline{\mathrm{V}}(\bar{f}(X_*))$

predict Algorithmus

Annahme: $y_i = f(x_i) + \varepsilon_i$, $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_n^2)$

Inputs: X (inputs), y (targets), σ_n^2 (noise),
K (covariance funtion), X_* (test input)

\# 1 $\quad L = \text{cholesky}(K(X, X) + \sigma_n^2 I)$
\# 2 $\quad \alpha = \text{solve}(L^\top, \text{solve}(L, y))$
\# 3 $\quad \bar{f}(X_*) = K(X, X_*)^\top \cdot \alpha$
\# 4 $\quad v = \text{solve}(L, K(X, X_*))$
\# 5 $\quad \overline{\mathrm{V}}(\bar{f}(X_*)) = K(X_*, X_*) - v^\top v$

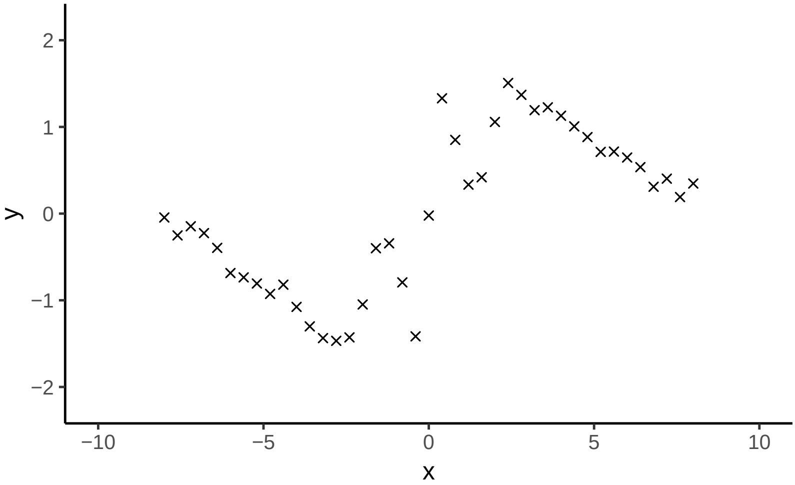
return: $\bar{f}(X_*)$, $\overline{\mathrm{V}}(\bar{f}(X_*))$

GPR

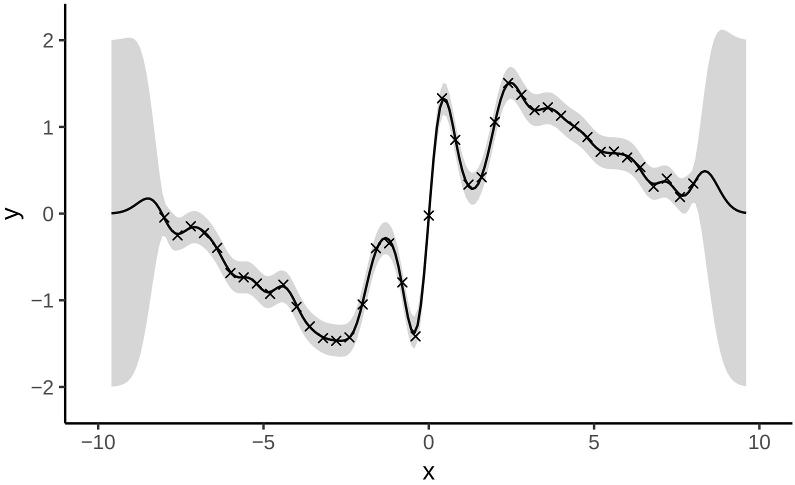
`GPR$new(X, y, noise, cov_func)`

- R6 Klasse
- Methoden
  - `$predict`
  - `$plot`
  - `$plot_posterior_draws`
  - `$plot_posterior_variance`
- Unterklassen für häufig auftretende Kovarianzfunktionen, die lediglich Parametereingabe erfordern

Regression



Regression



Optimierung der Hyperparameter

Suchen Kovarianzfunktion, die beobachtete Daten am besten erklärt

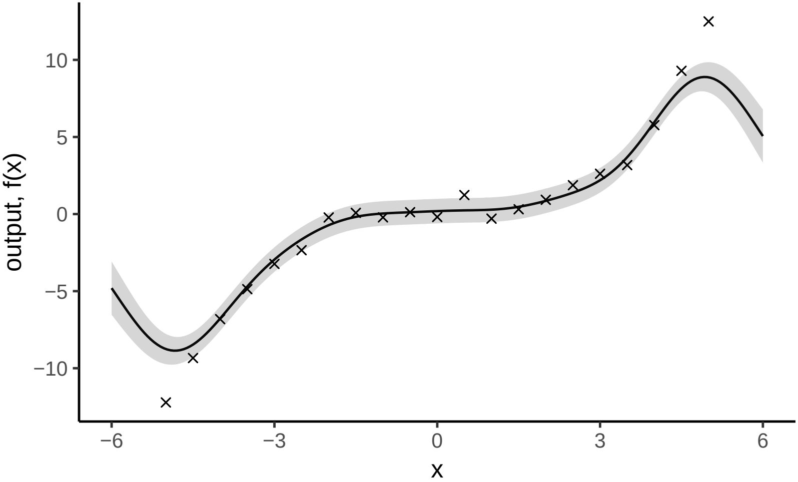
Methode: Maximiere Log-Likelihood der beobachteten Daten nach der Kovarianzfunktion k

fit()

```
fit(X, y, noise, cov_names)
```

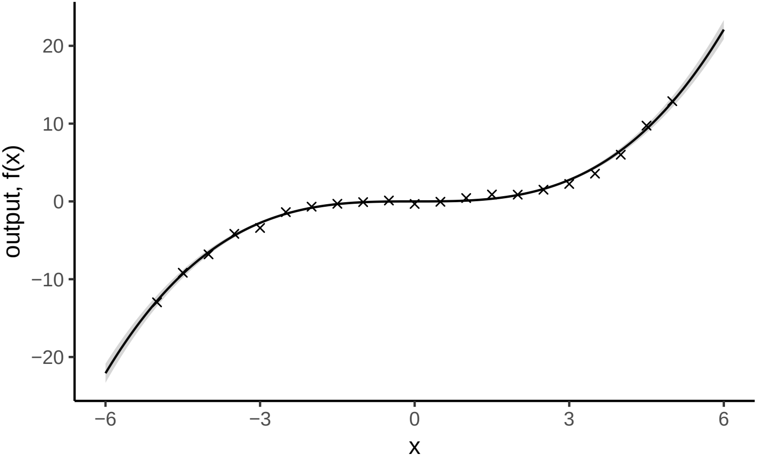
- Gibt die beste Kovarianzfunktion mit optimalen Parametern zurück
- Nutzen Newton-Methode, `optim()`
- Error handling bei numerischen Problemen der Cholesky Zerlegung

Gewählte Kovarianzfunktion




```
## The chosen covariance function is sqrexp
```

Optimierte Kovarianzfunktion



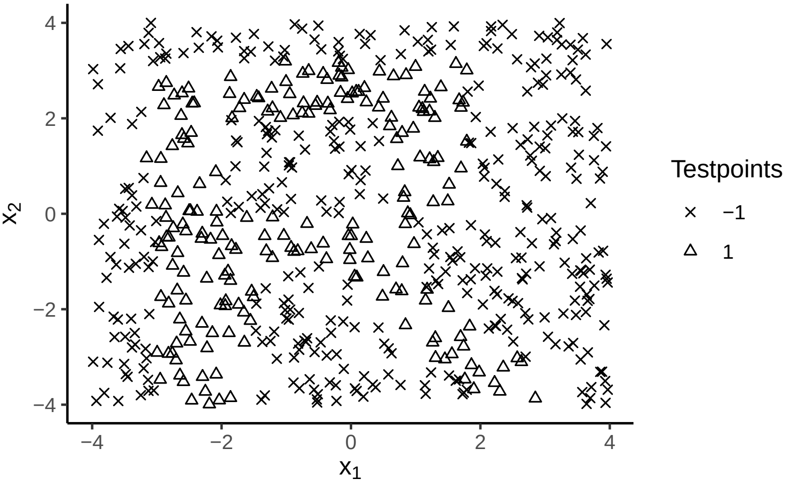

```
## The optimal covariance function is polynomial
```

GP Classification

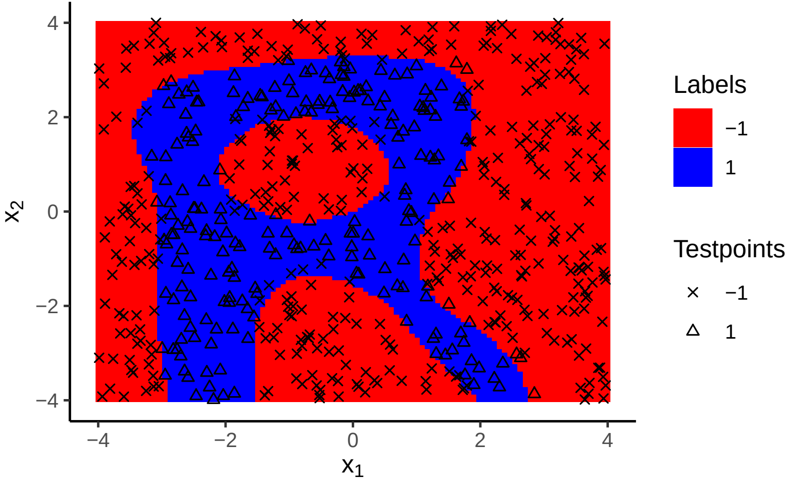
`GPC$new(X, y, cov_fun, epsilon)`

- R6 Klasse
- Methoden `$predict_class`, `$plot`
- Effizienz durch Vektorisierung

Klassifikation



Klassifikation

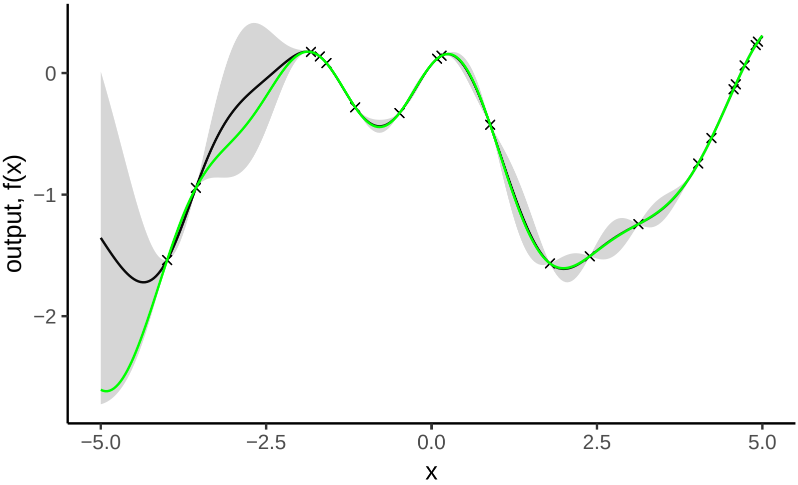


Simulation

Framework, um Qualität von GP Methoden für verschiedene Daten zu testen

- `simulate_classification(func, limits, training_size)`
- `simulate_regression(func, limits, training_size)`
- `simulate_regression_gp(actual_cov, limits, training_size)`

Simulation Beispiel



Shiny App