Gaussian Process Report

Budjan, Haas, Klumpp, Reitze

21 Februar 2019

Einleitung

Unser Paket implementiert Gaussian Process Regression (GPR), Classification (GPC) und eine Optimierungsfunktion der Hyperparameter für unser Regressionsmodell. Für die Regression haben wir eine übersichtliche Methode zur Visualisierung der Ergebnisfunktion mit Konfidenzbereichen sowie Methoden zur Visualisierung des zugrundeliegenden Gaußschen Prozesses erstellt. Für die Klassifikation existiert auch eine Methode, die einen zweidimensionalen Plot mit den Entscheidungsgebieten erzeugt. Des Weiteren haben wir ausführliche Simulationsfunktionen, die vorgegebene wie beliebige Beispiele visualisieren, und eine Shiny-App geschrieben, die Plots für ein breites Feld an Eingaben ausgibt und einen intuitiven Umgang mit unserer Regressionsklasse ermöglicht.

Regression

Wir haben uns dafür entschieden, eine R6-Klasse `GPR` für die Regression anzulegen. Grund hierfür war die leichte Erstellung von schreibgeschützen Bereichen. Dies erschien sinnvoll, da die Algorithmen einige rechenaufwendige Schritte beinhalten, die nur einmal beim Erstellen der Klasse ausgeführt werden sollten (z.B. Cholesky-Zerlegung). Der Nutzer soll davor geschützt werden beim Hinzufügen von Daten eine neue Berechnung der implizit genutzten Variablen in den Algorithmen zu vergessen. Obwohl unsere Klasse die Eingabe beliebiger Daten und Kovarianzfunktionen ermöglicht, besticht sie durch einfache und optimierte Nutzung, indem man nur die nötigen Inputs `X, y` und `noise` angibt. Eine Vektorisierung der von uns implementierten Kovarianz-Funktionen sowie der Vorhersage-Methode hat die Effizienz dieser Methode und damit auch die der Plots erheblich verbessert. Die Dokumentationen und die Vignette erläutern die einfache Nutzung unseres Pakets mit Minimalbeispielen. Die Unterklassen von `GPR` wurden im Laufe des Projekts überflüssig; ihre Nutzung ist Geschmackssache und bietet sich an, wenn man eine der vorimplementierten Kovarianzfunktionen mit bestimmten Parametern nutzen will.

Optimierung der Hyperparameter

Die Funktion zur Optimierung der Hyperparameter `fit(X ,y, noise, cov_names)` bestimmt die optimale Kovarianzfunktion unter den in `cov_names` als Liste Angegebenen. Sie beruht auf den Abschnitten zur Bayessian Model Selection in der gegebenen Quelle und maximiert die Log-Marginal-Likelihood der Trainingsdaten, modelliert als Gaußscher Prozess, als Funktion der Hyperparameter. Wir haben uns für diese Methode entschieden, da in der Quelle Cross-Validation als das numerisch aufwendigere Verfahren beschrieben wurde. Insbesondere ist bei letzterem die Invertierung der Ableitungsmatrix aufwändiger.

Hier haben wir Metaprogrammierung genutzt, um Kovarianzfunktionen mit unterschiedlich vielen Parametern zu optimieren. Die Möglichkeit, dass die Cholesky-Zerlegung der Matrix `K(X,X) + noise * diag(n)` numerisch instabil sein kann, wird in unserer Quelle nicht behandelt. Daher haben wir mit Error Handlern die Fehleranfälligkeit der Cholesky-Zerlegung von nahezu singulären Matrizen abgefangen, sodass für jede Kovarianzfunktion der optimale Parameter bis zum ersten Fehler zurückgegeben wird. So erzeugt `fit` stabile Ergebnisse, vor allem für `noise > 0` und Kovarianzfunktionen, die positiv definite Kovarianzmatrizen erzeugen.

Außerdem mussten wir sicherstellen, dass bei polynomiellen Kovarianzfunktionen die Exponenten aus der Menge der natürlichen Zahlen gewählt werden und somit über diskrete und stetige Parmeter optimieren. Dies haben wir gelöst, indem wir uns für den Exponenten auf die Menge $\{1, \ldots, 10\}$ beschränken und jeweils den

besten Wert für den zweiten Parameter sigma bestimmen. Dann wird die beste Kombination von Werten zurückgegeben.

Klassifizierung

Bei Klassifizierung sind wir ähnlich vorgegangen wie im Buch und wurden vor keine größeren Probleme gestellt. Wie bei der Regression haben wir die Prediction so weit wie möglich vektorisiert. Wegen des zu berechnenden Integrals ist die Prediction bei Klassifikation wesentlich langsamer als die Prediction bei Regression. Deshalb war die Beschleunigung durch Vektorisierung, vor allem beim Lösen der Gleichungssysteme, wichtig, um in kurzer Zeit gute Plots der Klassifikation erstellen zu können.

Plots

Für die Plot-Methoden der Klassen haben wir das Paket `ggplot2` verwendet. Damit lassen sich leicht mehrere Plots überlagern. Unsere Visualisierungen sprechen für sich. Sie erscheinen uns übersichtlich und vollständig.

Simulation

Die drei implementierten Simulationsfunktionen sind Frameworks, um Zusammenfassungen und Plots für verschiedene Eingaben zu erstellen, die einen schnellen Überblick über die Stärken und Schwächen der implementierten Algorithmen ermöglichen. Dabei erzeugt `simulate_regression_gp` eine Zufallsfunktion aus der vorgegebenen Kovarianzfunktion und den zugehörigen `fit`.

Shiny App

Die Shiny App stellt die wesentlichen Features des Pakets anhand von Plots dar. Wir haben hierbei die Möglichkeiten von `shiny` genutzt, um sowohl für Regression als auch für Klassifikation eine Visualisierung anzuzeigen. Außerdem haben wir durch eine dynamische Änderung des Interface die Möglichkeit erstellt für verschiedene Kovarianzfunktionen verschiedene Parameter anzupassen. Um die Ergebnisse der `fit`-Funktion darzustellen besitzt die App auch die Option die berechneten optimalen Parameterwerte zu verwenden. Durch Meta-Programmierung lassen sich beliebige Funktionen mit analytisch angebbarem Ausdruck verwenden. Der Plot zur Klassifikation lässt sich durch das manuelle Hinzufügen von Punkten verändern, um die Verschiebung der Entscheidungsregionen zu beobachten. Hierbei haben wir uns nur auf 2 dimensionale Probleme beschränkt, da deren Darstellung am anschaulichsten ist.

Eigenanteil

Abschließende Bemerkungen

Das Projekt war sehr zeitaufwendig. Dafür sind wir zufrieden mit unseren Ergebnissen und dass wir die meisten Inhalte der Vorlesung darin umsetzen und festigen konnten. Das Thema war spannend und unsere Gruppe hatte eine sehr konstruktive Arbeitsatmosphäre.