**6. МОДЕЛИ ПРОСТРАНСТВА СОСТОЯНИЙ ДЛЯ ВРЕМЕННЫХ РЯДОВ**

Модели пространства состояний аналогичны статистическим моделям, которые рассматривались ранее, но имеют большую “практическую” ценность. Они применяются при решении таких инженерных задач, как учет погрешности измерений и получение оценок на основе априорных знаний или убеждений.

Рассматриваемые здесь модели определяют системы, в которых истинное состояние не может быть измерено напрямую, но может быть выведено из результатов измерений. Модели пространства состояний основываются на знаниях о динамическом поведении системы, например, о временном развитии истинного состояния системы, определяемом внутренними процессами и внешними воздействиями.

Ранее вам, скорее всего, не доводилось математически рассчитывать модели пространства, хотя в повседневной жизни они встречаются повсеместно. Например, понаблюдаем за водителем автомобиля, “виляющим” в дорожном потоке. Постараемся определить его возможные маневры, чтобы избежать попадания с ним в ДТП. Если водитель находится под воздействием алкоголя, то вам следует обратиться в полицию, но если в его поведении не наблюдается повторяющихся шаблонов, то вам придется реагировать на возможные угрозы самостоятельно.

В следующие несколько секунд или минут вам нужно обновлять собственную модель пространства для этого водителя, чтобы понять, как правильно поступать в будущем.

Классический пример ситуации, описываемой моделью пространства состояний, — задача о запускаемой в космос ракете. Нам известны законы Ньютона, применяемые для описания динамики системы, и мы можем отследить движение ракеты в любой момент времени. Нам также известно, что GPS-датчики и другие инструменты отслеживания местоположения ракеты имеют некоторую погрешность измерения, которую можно определять количественно и нужно обязательно учитывать в выполняемых расчетах. Наконец, нужно понимать, что практически невозможно учесть все действующие на нашу ракету силы, поскольку система характеризуется большим количеством неизвестных параметров — нам нужно обеспечить ее устойчивость к самым разным источникам воздействия, таким как, например, солнечный ветер. Как оказалось, достижения в статистических и инженерных дисциплинах, накопленные за последние 50 лет, позволяют решать подобные задачи достаточно легко.

К разработке моделей пространства состояний и возникновению интереса к решаемым с их помощью задачам привели два разных исторических свершения. Во-первых, примерно в середине XX века мы вступили в эпоху механистической автоматизации. Человечеством были созданы ракеты и космические корабли, навигационные системы для подводных лодок и множество других средств автоматизации, работа которых основана на оценке состояния системы, которое не поддается измерению. Тогда исследователи задумались об оценке состояния системы с помощью специально разработанных инструментов анализа в пространстве состояний, позволяющих минимизировать ошибки измерения и другие виды неопределенности. В результате появились первые методы (анализа) пространства состояний. Во-вторых, параллельно с методами пространства состояний развивались технологии ведения учетной документации и соответствующих вычислительных алгоритмов.

Все это привело к получению гораздо более крупных наборов данных, используемых в качестве источников временных рядов, чем те, которые рассматривались нами ранее. Извлекаемые из них временные ряды имеют несравнимо большую длину, или плотность, временных точек. С увеличением количества доступных для изучения наборов данных возникла необходимость в разработке более информационно емких инструментов их обработки, предполагающих моделирование пространства состояний.

В этом разделе мы рассмотрим наиболее часто используемые методы пространства состояний.

• Фильтр Калмана для линейной гауссовой модели;

• Скрытые марковские модели;

• Байесовский структурный временной ряд.

Рассматриваемые методы хорошо описаны, доступны для реализации и обладают строго очерченной областью применения. Для каждого из них мы выработаем интуитивно понятную математическую модель и обсудим данные, подходящие для обработки с помощью того или иного метода. Наконец, мы приведем примеры кодов реализации методов пространства состояний.

В каждом случае важно научиться сопоставлять наблюдения с состояниями, в которых были проведены такие наблюдения. Оценивая основное состояние по наблюдениям, мы можем выделить следующие исследовательские процессы и этапы.

*Фильтрация*

Использование измерения в момент времени t для обновления оценки состояния в момент времени *t*.

*Прогнозирование*

Использование измерения в момент времени *t* - 1 для составления прогноза ожидаемого состояния в момент времени *t* (что позволяет также спрогнозировать измерение в момент времени *t*).

*Сглаживание*

Использование измерения в течение определенного периода времени, который включает в себя момент *t*, а также периоды до и после него, для оценки того, каково было истинное состояние в момент времени *t*.

Механика этих операций часто схожа, чего не скажешь о получаемых результатах. Фильтрация — это способ сопоставления последней информации с предыдущими данными при обновлении оценки состояния. Прогнозирование — это предсказание возможного состояния без получения сведений о будущем. Сглаживание — это использование информации о будущем и прошлом для получения наилучшей оценки состояния в данный момент времени.

***Модели пространства состояний: преимущества и недостатки***

Модели пространства состояний могут использоваться как в детерминированных, так и в стохастических приложениях и применяться как к непрерывным, так и к дискретным данным.

Уже одно это дает некоторое представление об их полезности и невероятной гибкости. Гибкость моделей пространства состояний предопределяет как достоинства, так и недостатки этого класса моделей.

У моделей пространства состояний много сильных сторон. Они позволяют моделировать наиболее интересные данные временных рядов: динамические процессы и состояния, которые генерируют анализируемые данные с шумом, а не одни только данные с шумом. С помощью модели пространства состояний вводится модель причинности в процесс моделирования, чтобы объяснить, что, в первую очередь, порождает процесс. Такой подход оказывается оправданным в случаях существования обоснованных теорий или надежных знаний о работе системы либо когда необходимо использовать модель для детального исследования общей динамики известной системы.

Модель пространства состояний позволяет изменять коэффициенты и параметры во времени, т.е. определять поведение системы во времени. Заметьте, что при использовании моделей пространства состояний условие стационарности на данные не накладывается. Это сильно отличается от ситуаций, рассматриваемых нами ранее, в которых предполагалось, что устойчивый процесс моделируется только одним набором не изменяющихся во времени коэффициентов.

Тем не менее у модели пространства состояний есть и ряд недостатков, а иногда они даже рассматриваются в качестве сильной составляющей модели.

• Поскольку модели пространства состояний очень гибкие, существует множество параметров, которые можно установить, и многие формы, которые они могут принимать. Это означает, что свойства конкретной модели пространства состояний часто оказываются недостаточно изученными. При построении модели пространства состояний, адаптированной к исследуемым данным временных рядов, вы вряд ли найдете учебники по статистике или научные статьи, в которых она уже рассматривалась. Таким образом, вы оказываетесь в менее определенной ситуации, пытаясь понять, как работает модель, или определить, где были совершены ошибки.

• Модели пространства состояний могут быть очень сложными в вычислительном отношении, поскольку включают много параметров. Кроме того, слишком большое количество параметров для некоторых типов моделей пространства состояний может сделать вас зависимым от переобучения, особенно при недостаточности данных.

***Фильтр Калмана***

Фильтр Калмана — это хорошо исследованный и популярный метод для включения новой информации из временного ряда и ее разумного объединения с ранее известной информацией для оценки основного состояния. Одно из первых применений фильтра Калмана произошло во время миссии Apollo 11: когда инженеры НАСА поняли, что встроенные вычислительные элементы не позволят использовать другие, более ресурсоемкие методы оценки положения, они выбрали этот фильтр. Как вы увидите в этом разделе, преимущества фильтра Калмана заключаются в том, что его относительно легко вычислить, и он не требует хранения прежних данных для составления текущих оценок или будущих прогнозов.

**Обзор**

Вычислительные методики, применяемые для описания фильтра Калмана, способны озадачить начинающих специалистов по обработке данных — не только исходя из их высокой сложности, но и потому, что в них приходится отслеживать значительное количество величин — это итеративный, отчасти замкнутый процесс со многими взаимосвязанными величинами. По этой причине здесь мы не будем выводить уравнения фильтра Калмана, а всего лишь приведем общее их описание, чтобы понять, как они работают (настоятельно рекомендуется познакомиться с многочисленными альтернативными описаниями фильтра Калмана на сайте Mathematics StockExchange (https://perma.сс/27RK-YQ52)).

Начнем с изучения линейной гауссовой модели, в которой утверждается, что состояние и наблюдения имеют следующую динамику:



Как видите, состояние в момент времени *t* является функцией состояния на предыдущем временном шаге , внешнего воздействия  и стохастической составляющей . Аналогично измерение в момент времени *t* является функцией состояния в момент времени *t* и члена стохастической ошибки, т.е. ошибки измерения.

Давайте представим, что  — это реальное положение космического корабля, а  — положение, измеряемое с помощью некого измерительного устройства (датчика). Пусть  — ошибка измерения такого датчика. Тогда основное уравнение фильтра Калмана будет показывать обновленную оценку с учетом новой информации для времени *t*: 

Здесь мы переходим к этапу фильтрации, на котором принимается решение о том, как измерение в момент времени *t* влияет на обновленную оценку состояния в момент времени *t*. Не забывайте, что мы рассматриваем ситуацию, в которой выступаем простыми наблюдателями, и, делая выводы о состоянии, не можем быть в них уверены. Выше показано, что величина  задает баланс между старой информацией  и новой информацией нашей оценки.

Чтобы перейти к более подробному описанию, нам нужно определиться с используемой терминологией. Величина обозначает оценки ковариации нашего состояния (это может быть скаляр или матрица, в зависимости от размерности состояния — многомерные состояния более распространены).  — это оценка для *t* до учета измерения в момент времени *t*.

Кроме того, величиной *R* мы будем определять дисперсию ошибки измерения, т.е. дисперсию  которая также может представляться либо скаляром, либо ковариационной матрицей в зависимости от размерности измерений. Как правило, в реальных системах параметр *R* хорошо известен, поскольку описывает известные физические свойства конкретного датчика или измерительного устройства.

Соответствующее ему значение , которое выражается через *Q*, определено менее точно и подлежит уточнению в процессе моделирования.

Начать исследование стоит с процесса, в котором известны или оцениваются значения *х* и *Р* в момент времени 0. Продвигаясь вперед по временной шкале, мы будем последовательно чередовать фазы прогнозирования и обновления так, чтобы каждая следующая фаза прогнозирования предшествовала последующим этапам обновления/фильтрации и т.д.

*Прогноз:*

**

*Фильтрация:*

**

где *Kt* матрица коэффициентов усиления фильтра Калмана равна

**.

Существует много вариантов визуализации такого рекурсивного процесса.

Иногда его разбивают на множество этапов (чаще всего — на четыре или пять). Однако самый простой способ его описания состоит в получении прогнозных значений в момент времени *t* без измерения  (прогноз) и проведения вычислительных этапов для момента времени *t* уже после того, как измерение  станет известным (фильтрация).

Для выполнения таких действий нам понадобятся следующие значения.

• Оценки для *R* и *Q* — ковариационные матрицы значений ошибок измерения (легко вычисляются) и стохастичности состояния (обычно оцениваются) соответственно.

• Оценки или известные значения состояния в момент времени *0* (оценивается по значению ).

• Представление о том, какие силы будут воздействовать на систему в момент времени *t* и как они повлияют на состояние, т.е. матрица *В* и значение 

• Представление о динамике системы, определяющей переход состояний от одного временного шага к другому, а именно *F*.

• Понимание зависимости измерения от состояния системы, а именно положения.

Существует много способов получения уравнений фильтра Калмана, в том числе с вероятностной точки зрения в терминах математического ожидания, минимизации наименьших квадратов или оценки максимального правдоподобия.

Все они широко освещены в специализированной литературе, и вы легко найдете их, выполнив поиск на тематических сайтах в Интернете.

***Код реализации фильтра Калмана***

Рассмотрим классический вариант использования фильтра Калмана: попробуем отследить объект, подчиняющийся законам Ньютона, с помощью датчиков, передающих показания с некой ошибкой. Сгенерируем временные ряды, основываясь на ньютоновских законах движения тел, согласно которым положение объекта является функцией его скорости и ускорения. Несмотря на непрерывность физического процесса, движение будет отслеживаться по данным дискретных измерений. В самом начале создадим ряд со значениями ускорения, а затем предположим, что начальные положение и скорость объекта равны *0*. Хотя это не совсем реалистично, будем предполагать, что ускорение изменяется мгновенно в начале каждого временного шага и остается постоянным в течение всей его длительности.

## R

## Ракета движется 100 временных шагов

ts.length <- 100

## Движение является ускоренным

а <- rep(0.5, ts.length)

## Начальное положение и скорость равны 0

х <- rep(0, ts.length)

v <- rep(0, ts.length)

for (ts in 2:ts.length) {

x[ts] <- v[ts - 1] \* 2 + x[ts - 1] + 1/2 \* a[ts - 1]^ 2

x[ts] <- x[ts] + rno:rm(l, sd = 20) ## стохастическая компонента

v[ts] <- v[ts - 1] + 2 \* a[ts - 1]

}

Если вы не помните законы Ньютона, то обязательно повторите их, чтобы не принимать на веру все дальнейшие выкладки (в первую очередь, касающиеся вычисления значений х [ts ] и v [ts ]).

Динамические характеристики движения, которое описывается заданными ранее параметрами, показаны на графиках, изображенных на рисунке.

## R

Par (mfrow = с(3, 1))

plot (х, main = "Положение",type='1')

plot (v, main = "Скорость", type='1')

plot (acceleration, main = "Ускорение", type='1')

Предполагается, что переменные полностью описывают состояние, но единственные доступными данные — это сведения о положении объекта, и они поступают с зашумленного датчика. В следующем коде положение такого датчика задается переменной х, а измеренные значения соотносятся с фактическими данными о положении объекта так, как показано на рисунке.

## R

z <- х + rnorm(ts.length, sd = 300)

plot(x, ylim = range (c(x, z)))

lines(z)

Как видно на рисунке, движение характеризуется постоянным ускорением (нижний график), обеспечивающим линейное увеличение скорости (средний график) и изменение координаты по параболической траектории (верхний график).

Если такое поведение объекта кажется вам непонятным, то либо примите его как должное, либо повторите курс механики в любом учебнике по физике. Применим фильтр Калмана. Сначала напишем общую функцию, отражающую параметризацию системы и результаты, полученные ранее в этом разделе.

## R

kalman.motion <- function(z, Q, R, A, H) {

dimState = dim(Q) [1]

xhatminus <- array(rep(0, ts.length \* dimState),

c(ts.length, dimState))

xhat <- array (rep (0, ts.length \* dimState),

c(ts.length, dimState))

Pminus <- array (rep (0, ts.length \* dimState \* dimState),

c (ts.length, dimState, dimState))

P <- array (rep (0, ts.length \* dimState \* dimState),

c(ts.length, dimState, dimState))

К <- array(rep(0, ts.length \* dimState),

c(ts.length, dimState)) # коэффициент усиления Калмана

# Начальное предположение о равенстве нулю всех параметров

xhat[1, ] <- rep(0, dimState)

P[l, , ] <- diag(dimState)

for (k in 2:ts.length) {

# Обновление времени

xhatminus[k, ] <- A %\*% matrix (xhat [k - 1, ])

Pminus[k, , ] <- A %\*% P[k - 1, , ] %\*% t(A) + Q

K[k, ] <- Pminus[k, , ] %\*% H %\*%

solve (t(H) %\*% Pminus[k, , ] %\*% H + R)

xhat[k, ] <- xhatminus[k, ] + K[k, ] %\*%

(z[k]- t(H) %\*% xhatminus[k, ])

P[k, , ] <- (diag(dimState)-K[k,] %\*% t(H)) %\*% Pminus[k, , ]

}

## Возвращение прогноза и сглаженного значения

return (list (xhat = xhat, xhatminus = xhatminus) )

}

Теперь применим эту функцию для определения одного только положения (но не ускорения и скорости) ракеты.

## R

## Параметры шума

R <- 10^2 ## Дисперсия измерений. Предопределяется физическими

## ограничениями, накладываемыми на измерительное

## оборудование. Согласуется с шумом, добавленным к х,

## в представленном выше коде

Q <- 10 ## Дисперсия процесса. Обычно это гиперпараметр,

## обеспечивающий наибольшую точность

## Динамические параметры

А <- matrix (1) ## x\_t = А \* x\_t-l (как предыдущее значение х влияет

на последующее значение х)

Н <- matrix(1) ## y\_t = Н \* x\_t (перевод состояния в измерение)

## Прогонка данных через фильтр Калмана

xhat <- kalman.motion(z, diag(1) \* Q, R, A, H) [ [1] ]

Представим реальные, измеренные и прогнозируемые положения на общем графике.

Фильтр Калмана удаляет большую часть шума из ошибки измерения. Насколько много — зависит от значения *R*, параметра шума измерения, который отражает способность фильтра взвешивать последнее значение по сравнению с более ранними значениями. Как видим, фильтр вполне удовлетворительно справляется с прогнозированием данных. В частности, между прогнозными и фактическими данными не наблюдается запаздывание, а это указывает, что прогноз текущего значения строится на основе последнего значения.

Мы рассмотрели самый простой пример фильтра Калмана. Он хорошо исследован и находит широкое применение в различных приложениях, особенно в системах с неплохо изученной внутренней динамикой. Это делает его идеальным инструментом для решения задач, подобных простому движению ракеты, с достаточно точно изученными процессами изменения системы.

Обратите внимание на то, что в этом простом примере возможности и преимущества фильтра Калмана раскрыты не полностью. В общем случае он оказывается полезным при решении задач с несколькими типами измерений — разных величин или параллельных измерений одного и того же показателя несколькими устройствами. Существует множество вариантов фильтра Калмана, имеющих большое прикладное значение в самых разных дисциплинах и областях знаний. Как было показано выше, одно из главных преимуществ фильтра Калмана заключается в его рекурсивности. Это устраняет необходимость в просмотре всех предыдущих точек данных в каждой итерации процесса. Скорее, на каждом временном шаге вся информация из предыдущих временных шагов оптимальным образом включается в несколько оценочных параметров, а именно — в последние оценки состояния и ковариации. К достоинствам метода стоит отнести взвешенное обновление данных, когда сводные статистические значения исследуемых показателей оптимально сопоставляются только с последними данными.

Такой подход делает фильтр Калмана наиболее востребованным в приложениях, где наибольшую ценность представляют вычислительная производительность и скорость обработки данных. Во многих случаях он прекрасно описывает динамику реальных систем, представляемых с помощью Марковских процессов (хранящими сведения только о предыдущих состояниях) и функции базового состояния, которое может быть измерено только с некоторой ошибкой.

Существует много полезных модификаций рассмотренного выше фильтра Калмана. Один из наиболее распространенных его вариантов отличается адаптацией к сглаживанию — возможностью получить наилучшую оценку истинного состояния в момент времени *t* на основе данных как до, так и после времени *t*. Математические уравнения и код реализации такого подхода во многом схожи с представленными выше. Еще один вариант — расширенный фильтр Калмана (Extended Kalman Filter — EKF), в котором фильтр Калмана адаптируется к данным процессов с нелинейной динамикой. Его достаточно просто реализовать благодаря включению специальных инструментов в различные пакеты языков R и Python.

Сложность фильтра Калмана составляет *О*(*Т*) относительно длины временного ряда и *O*(*d*2) — относительно *d*, размерности состояния. Это указывает на то, что нужно отказываться от переопределения состояния в случаях, когда более упорядоченная спецификация уже работает хорошо. Однако именно линейность по отношению к длине временных рядов делает фильтр Калмана широко используемым в реальных производственных задачах и гораздо более популярным, чем другие фильтры, предназначенные для моделирования пространств состояний временных рядов.

***Скрытые марковские модели***

Скрытые марковские модели (Hidden Markov Models — HMM) — это особенно полезное и интересное средство моделирования временных рядов, представляющее в анализе временных рядов редкий случай обучения без учителя, в котором обучение ведется в отсутствие обозначенного правильного ответа. Модели НММ обосновываются исходя из интуитивных соображений, подобных используемым при описании фильтра Калмана ранее в этой главе, — предположении о том, что наблюдаемые переменные могут быть не самыми информативными для рассматриваемой системы. Как и в случае фильтра Калмана, примененного к линейной гауссовой модели, будем полагать, что процесс имеет состояния, а наблюдения предоставляют информацию о таких состояниях. Как и прежде, нам нужно получить представление о том, как переменные состояния связаны с проведенными наблюдениями. В случае применения НММ утверждается, что процесс нелинейный и характеризуется скачками между дискретными состояниями.

***Обзор***

Модель НММ описывает систему с непосредственно не наблюдаемыми состояниями.

Система описывается Марковским процессом, не хранящим сведений обо всех предыдущих состояниях — при расчете вероятности будущих событий учитываются только текущие состояния системы. Таким образом, сведения о предыдущих состояниях системы оказываются менее полезными, чем знание текущего состояния системы.

Марковские процессы часто описываются в матричном изложении. Например, рассмотрим систему, которая характеризуется состояниями А и В, между которыми возможны переходы. В любом из состояний система с большей статистической вероятностью будет оставаться в прежнем состоянии, а не переходить в другое состояние на любом отдельном временном шаге. Такая система может описываться следующей матрицей вероятностей.

|  |  |  |
| --- | --- | --- |
|  | А | В |
| А | 0,7 | 0,3 |
| В | 0,2 | 0,8 |

Представим, что наша система находится в состоянии А, а именно — (1, 0). (Состояние В описывается как (0, 1).) В таком случае вероятность того, что система останется в состоянии А, составляет 0,7, тогда как вероятность изменения состояния равна 0,3. Заметьте, что здесь совершенно не важно, в каких состояниях система находилась до последнего момента времени. Именно такие процессы называются Марковскими.

Скрытая Марковская модель представляет собой систему такого же типа, за исключением того, что в ней нельзя напрямую сделать вывод о состоянии системы, исходя из наблюдений. Вместо этого наблюдения выступают подсказками в предположениях о состоянии системы (рисунок).

*q*(*t*-1) 

*q*(*t*) 

*q*(*t*+1) 

*y(t*-1) 

*y*(*t*) 

*y*(*t*+1) 

Обратите внимание на то, что в действительных приложениях состояния обычно создают перекрывающиеся выходы, поэтому далеко не всегда понятно, какое из состояний за какой выход отвечает. Например, мы собираемся применить модель НММ к данным, подобным представленным на рисунке.



Это данные, смоделированные для системы с четырьмя состояниями, но простой анализ графика временных рядов не позволяет определить количество состояний, их границы и области переходов.

Тем не менее модели НММ находят применение в следующих прикладных задачах.

• Определение смены режима финансовых рынков (<https://perma.cc/> JRT2-ZDVJ).

• Классификация, прогнозирование и исправление данных в последовательностях ДНК (https: //perma. cc/4V4A-53TZ).

• Распознавание стадий сна по данным ЭКГ (<https://perma.cc/G37YXBQH>).

***Обучение модели***

Мы утверждаем, что существует состояние, которое невозможно измерить напрямую, и во многих наборах данных, к которым можно было бы применить метод, невозможно определить их визуально. Тогда как нужно действовать, чтобы распознать скрытые состояния, не обладая априорными знаниями о них? Ответ: проходом по состояниям. Не существует волшебной палочки для получения наиболее вероятной последовательности скрытых состояний для объяснения наблюдений, но всегда можно провести их оценку, правильно описав систему.

В модели НММ утверждается, что система полностью описывается при наличии следующей информации.

• Вероятность перехода от *х*(*t*) к *x*(*t* + 1). Задается с помощью матрицы, аналогичной описанной выше и устанавливающей вероятности переходов между состояниями А и В. Размер такой матрицы зависит от количества гипотетических состояний.

• Вероятность эмиссии, или вероятность наблюдения *y*(*t*) при заданном значении *x*(*t*).

• Начальное состояние системы.

Рассмотрим конкретный случай, в котором обозначим переменные, принимающие участие в определении и обучении НММ-процесса.

• Q = *q*1, *q*2, ..., *q*N различных состояний системы.

• А = *аi,j* = *а1,1*, *а1,2*, ..., *aN,N* — матрица вероятностей перехода, определяющая переход на любом заданном временном шаге из состояния *i* в состояние *j.*

• О = *o1, о2, ..., оТ* — последовательность наблюдений, отобранных из этого процесса в порядке следования, т.е. временного ряда наблюдений.

• *bj(ot)* (вероятность эмиссии (вероятность наблюдения данного значения, *оt* если состояние *qj).*

• p = *p1, p2,…,pN* - начальные распределения вероятностей, а именно — вероятность того, что система имеет начальное состояние *q1, q2,…,qN* соответственно.

Однако в реальных данных обычно ни одна из этих переменных не определена. Чаще всего изначально известна только фактическая последовательность наблюдаемых значений *y1, y2,…,yt*.

***Алгоритм Баума-Уэлча***

Для оценки параметров скрытой марковской модели воспользуемся алгоритмом Баума-Уэлча. Он сводится к решению сложной задачи — оценке значений всех параметров, подробно описанных в предыдущем разделе. Это очень многогранная задача. Разобьем ее на следующие подзадачи.

• Определение вероятностей эмиссии для каждого возможного скрытого состояния и перехода из каждого возможного скрытого состояния в другое возможное скрытое состояние. Используем алгоритм Баума-Уэлча.

• Определение наиболее вероятного скрытого состояния на каждом временном шаге с учетом всей истории наблюдений. Используем алгоритм Витерби (описан ниже). Это родственные задачи, каждая из которых достаточно сложна и требует для решения больших вычислительных затрат. Более того, они связаны одна с другой. Для решения двух взаимосвязанных задач — оценки параметров и максимизации вероятности — можно использовать алгоритм максимизации математического ожидания для перехода между этими двумя этапами, пока не будет найдено приемлемое решение.

Чтобы применить алгоритм Баума-Уэлча, первым делом нужно определить функцию правдоподобия, которая представляет вероятность наблюдения имеющейся последовательности при заданных гипотетических параметрах. В нашем случае такими предполагаемыми параметрами будут математические параметры для каждого постулируемого состояния.

Например, предположим, что состояния описываются гауссовыми выходными данными с различными средними значениями и стандартными отклонениями в наблюдаемых значениях, зависящими от состояния. Рассмотрим модель с двумя такими состояниями, которую будем описывать в терминах *μ*1, *σ*1 и *μ*2, *σ*2, где *μu=i* обозначает среднее значение *i*-го состояния, *σi* — стандартное отклонение *i* –го состояния. С их помощью рассчитываются вероятности эмиссии, все вместе обозначаемые как *θ*. Кроме того, нам нужно обозначить последовательность состояний *x1, x2,…,xt* (все вместе *Xt*), которые нами не наблюдаются, но давайте представим, что они все же существуют.

Функция правдоподобия такой системы должна описывать вероятность наблюдения существующей последовательности при заданных параметрах вероятности эмиссии (т.е. вероятности наблюдения в строго заданном состоянии) и последовательности скрытых состояний как интеграла по всем возможным *Xt* таких что:



Однако это сложная задача по нескольким причинам, включая факт ее экспоненциального усложнения с увеличением количества временных шагов, что указывает на невозможность проведения исчерпывающего анализа по перекрывающимся значениям. Следовательно, нам нужно упростить задачу, обратившись к следующему ***ЕМ-алгоритму***.

1. Произвольно инициализируем переменные вероятности эмиссии.

2. Вычисляем вероятность каждого возможного *Xt*, учитывая значения вероятности эмиссии.

3. Используем эти значения *Xt*, чтобы получить лучшую оценку переменных вероятности эмиссии.

4. Повторяем пп. 2 и 3 до тех пор, пока не будет достигнута сходимость.

Более неформально это означает, что при случайном размещении двух распределений мы будем рассматривать каждый временной шаг и определять на каждом из них вероятность наблюдения определенного состояния (например, состояния А или В на временном шаге *t*). Назначив предполагаемое состояние для каждого временного шага, используем эти метки для переоценки вероятностей эмиссии (обнуляя лучшие среднее и стандартное отклонение для состояния). Затем процесс нужно повторить заново, используя недавно обновленные переменные вероятности эмиссии для улучшения оценки траектории *Xt.*

Не забывайте, что найти оптимальный набор параметров в таком способе использования ЕМ-алгоритма удается далеко не всегда. Может понадобиться много попыток, прежде чем вы добьетесь успеха, — продолжительность операции зависит от конкретных данных и выбранной модели.

***Алгоритм Витерби***

После оценки параметров процесса НММ, например, с помощью алгоритма Баума - Уэлча, нужно выполнить следующую, не менее важную задачу, заключающуюся в определении наиболее вероятного ряда состояний, основанных на временном ряду наблюдаемых значений.

В отличие от алгоритма Баума-Уэлча алгоритм Витерби гарантированно предоставляет наилучшее решение поставленной выше задачи. А все потому, что он относится к алгоритмам динамического программирования, предназначенным для полного и эффективного изучения диапазона возможных совпадений методом сохранения решений, полученных для отдельных участков пути, что позволяет отказаться от повторного пересчета всех возможных вариантов в случае удлинения пути.

***Динамическое программирование***

Специалисты по анализу данных не всегда располагают полным набором алгоритмов, но при изучении временных рядов точно не смогут обойтись без базовых решений, в которых предположения и повторное рассмотрение выступают необычайно важными техниками исследования упорядоченных во времени данных.

Проще всего объяснить, что такое динамическое программирование, на уже ставшем классическим примере последовательности чисел Фибоначчи. Представьте, что вам требуется вычислить восьмое число Фибоначчи. Проще всего получить его, зная шестое и седьмое числа последовательности Фибоначчи. А чтобы получить их, нужны четвертое и пятое числа Фибоначчи и т.д. Таким образом, решение более сложных задач по вычислению последовательностей Фибоначчи основывается на результатах более простых таких задач. Следовательно, при вычислении чисел Фибоначчи их нужно как-то сохранять, чтобы иметь возможность использовать в последующих, еще более сложных задачах. По этой причине динамическое программирование также известно как мемоизация.

Динамическое программирование применяется для решения задач со следующими отличительными признаками.

• Решение задачи размера *N* можно основать на решении задачи размера *N* - 1. По этой причине мемоизация решений более ранних, простых задач помогает более эффективно решать более поздние, заведомо более сложные задачи.

• Задачи имеют явно выраженный порядок масштабирования от более простой к более сложной.

• Всегда можно выделить базовую задачу, подлежащую простому расчету.

***Код обучения модели НММ***

Хотя процесс обучения модели НММ очень сложен, он реализуется с помощью специализированных пакетов языка R очень просто. В следующем примере используются инструменты пакета depmixS4. Вначале нужно получить надлежащий временной ряд. Воспользуемся следующим кодом.

## R

## Обратите внимание: здесь задается начальное значение

## При одном и том же начальном значении числа должны совпадать

set. seed (123)

## Параметры распределения для всех четырех состояний рынка,

## подлежащих моделированию

bull\_mu <- 0.1

bull\_sd <-0.1

neutral\_mu <- 0.02

neutral\_sd <- 0.08

bear\_mu <- -0.03

bear\_sd <- 0.2

panic\_mu <- -0.1

panic\_sd <- 0.3

## Представление параметров векторами для упрощения индексации

mus <- c(bull\_mu, neutral\_mu, bear\_mu, panic\_mu)

sds <- c(bull\_sd, neutral\_sd, bear\_sd, panic\_sd)

## Константы, описывающие генерируемые временные ряды

NUM.PERIODS <- 10

SMALLEST.PERIOD <- 20

LONGEST.PERIOD <- 40

## Определение рыночных дней случайным образом.

## Каждый такой день соответствует отдельному состоянию рынка

days <- sample (SMALLEST. PERIOD:LONGEST. PERIOD, NUM. PERIODS,

replace = TRUE)

## Генерирование временного ряда состояний рынка для заданного

## количества дней и его добавление к общему временному ряду

returns <- numeric ()

true.mean <- numeric()

for (d in days) {

idx = sample(l:4, 1, prob = c(0.2, 0.6, 0.18, 0.02))

returns <- c (returns, rnorm(d, mean = mus [idx], sd = sds [idx]))

true.mean <- c(true.mean, rep(mus [idx], d))

}

В предыдущем коде моделируется процесс биржевой торговли на бычьих, медвежьих, нейтральных и панических рынках. В нем устанавливается случайное количество дней, для которых сохраняются состояния рынка, а также определяются переменные, описывающие распределение вероятности эмиссии для каждого состояния (\_mu и sd\_, хранящие значения, которые подлежат измерению в заданном состоянии).

Чтобы получить представление о сгенерированном кодом временном ряде и частотности каждого состояния, нужно понять, сколько дней в выборке соответствует

каждому значению переменной true.mean, по которой отслеживаются состояния.

## R

table (true .mean)

Невероятно, но факт! Несмотря на намерение включить четыре состояния в моделируемый ряд, их было добавлено только три. Скорее всего, это связано с очень низкой вероятностью включения (0,02) четвертого состояния. Мы видим, что наименее вероятное состояние даже не было выбрано для добавления в ряд. Таким образом, далеко не всегда известно, что для заданного временного ряда учитываются не все возможные состояния, что еще раз свидетельствует о высокой сложности алгоритма модели НММ и сложности его обучения. Как бы там ни было, дальнейший анализ будет выполнен для группы из четырех состояний, чтобы увидеть, к какому результату это приведет (обратите внимание, что еще одна проблема с моделируемыми данными заключается в том, что мы не создали матрицу вероятности перехода из одного состояния в другое для управления потоком скрытого состояния. По сути, мы предположили, что состояние с большей вероятностью останется таким, как есть, в течение многих дней подряд, а затем с равной вероятностью сменится любым другим состоянием. Мы упустили формальную спецификацию и отказались от использования матрицы переходов для упрощения кода).

Можно переходить к обучению НММ. Результирующая модель НММ будет представлять временные ряды апостериорных вероятностей для каждого состояния для любого количества указанных состояний. В соответствии с более ранним описанием ЕМ-алгоритма от нас требуется указать всего одно значение — число предполагаемых состояний. Остальные параметры будут определяться по мере выполнения вычислительных этапов.

Как это часто бывает, при работе с программными пакетами наиболее сложная часть анализа на самом деле оказывается очень простой в реализации и представлении в коде. В нашем случае используется пакет depmixS44 (название этого пакета заимствовано у альтернативного названия “НММ” — Dependent Mixture Models (Модель зависимых смесей)) языка R. Обучение модели проводится дважды. Во-первых, с помощью функции depmix(), в которой устанавливается ожидаемое распределение, указываются число состояний и входные данные, которые будут использоваться при обучении. Далее в действие вступает функция fit(), которая принимает в качестве входных данных спецификацию модели. И только после этого применяется функция posterior(), генерирующая апостериорное распределение подписей состояний с учетом соответствия данных. Начиная с этого момента модель считается обученной, и нам остается только решить задачу по разметке данных, чтобы оценить параметры, описывающие распределения состояний и вероятности переходов.

## R

require(depmixS4)

hmm.model <- depmix(returns ~ 1, family = gaussian(),

nstates = 4, data=data.frame(returns=returns))

model.fit <- fit(hmm.model)

post\_probs <- posterior (model.fit)

Этим кодом создается модель hmm.model, в которой в качестве наблюдаемого указывается вектор returns. В нем определяется число состояний (4) и соответствие вероятности эмиссии нормальному распределению (см. параметр family). Модель обучается с помощью функции fit(), а для вычисления апостериорных вероятностей применяется функция posterior(). Апостериорные вероятности определяют правдоподобие состояний в данное время для параметров модели, определенных в процессе обучения.

Теперь можно переходить к визуализации состояний на одном графике с измеренными значениями.

## R

plot (returns, type = '1', lwd = 3, col = 1,

yaxt = "n", xaxt = "n", xlab = " ", ylab =" ",

ylim = c(-0.6, 0.6))

lapply(0:(length(returns) - 1, function (i) {

## Добавление прямоугольника соответствующего цвета,

## обозначающего состояние в заданный момент времени

rect (i,-0.6, (i + 1),0.6,

col = rgb(0.0,0.0,0.0,alpha=(0.2 \* post\_probs$state[i + 1])),

border = NA)

}

Сведения о предполагаемых параметрах распределения хранятся в отдельных атрибутах. При их просмотре помните об исходных настройках генерации данных.

Bull\_mu <- 0.1

bull\_sd <- 0.1

neutral\_mu <- 0.02

neutral\_sd <- 0.08

bear\_mu <- -0.03

bear\_sd <- 0.2

panic\_mu <- -0.1

panic\_sd <- 0.3

Попытавшись сравнить состояния, фактически обозначенные в данных (режим паники на бирже исключен из данных), можно получить представление о корреляции между следующими группами.

attr(model.fit,"response")

[ [1] ]

[[1]][[1]] <- среднее значение близко к режиму паники, но этот режим в выборке не представлен, поэтому четвертому состоянию назначены большие отрицательные значения

Model of type gaussian (identity), formula: returns ~ 1

Coefficients:

(Intercept)

-0.09190191

sd 0.03165587

[[2]]

[ [2] ] [ [1] ] <- соответствует режиму медвежьего рынка

Model of type gaussian (identity), formula: returns - 1

[[3]]

[[3]] [[1]] <- соответствует режиму бычьего рынка

Model of type gaussian(identity), formula: returns - 1

[[4]]

[[4]][[1]] <- соответствует режиму нейтрального рынка

Model of type gaussian (identity), formula: returns - 1

Одна из возможных причин, по которым обучение не привело к хорошему согласованию с базовыми скрытыми состояниями, заключается в отказе от использования правильной матрицы переходов, необходимой для обучения. В результате переходы между состояниями не были Марковскими, и это оказало пагубное влияние на обучение. Кроме того, мы пытались приблизить относительно короткий временной ряд с небольшим количеством переходов между состояниями, в то время как модели НММ лучше работают на более длинных временных рядах с большим количеством переходов. Я бы рекомендовал придумать более реалистичные искусственные данные для тестирования предлагаемой модели НММ. Помните, что в большинстве практических задач приходится работать с ненаблюдаемыми состояниями, поэтому перед тем как взяться за более амбициозные проекты, постарайтесь разобраться в факторах, вносящих ограничения в точность модели, в предельно простых случаях (с искусственными данными).

Модели НММ подходят для анализа многих видов данных. Они использовались для моделирования поведения финансовых рынков в фазе роста и рецессии, определения стадии скручивания белка в клетках и описания перемещения людей (до появления глубокого изучения). На сегодняшний день они остаются востребованными — чаще в задачах исследования динамического поведения систем, чем прогнозирования. Кроме того, модели НММ предоставляют больше сведений о процессах, чем точечные оценки или прогнозы. В такую модель легко включить априорные знания или предположения, например, указав количество состояний, используемых для ее обучения. Тем самым обеспечиваются преимущества статистических методов, но сохраняется возможность параметризации априорных знаний о системе.

Математические принципы и уравнения, используемые для расчета моделей НММ, хорошо изучены и доступны для понимания. На ваше рассмотрение предлагается большое количество программных инструментов и численных алгоритмов оптимизации, используемых при обучении моделей НММ по данным. Вы также познакомитесь с методами динамического программирования, с которыми нужно быть на “ты” каждому специалисту по анализу данных или программисту.

Как и фильтры Калмана, модели НММ могут использоваться для решения задач самых разных типов. На самом деле количество задач логического вывода, связанных с НММ-системами, увеличивается с повышением сложности дискретных состояний, каждое из которых имеет собственную вероятность эмиссии. Перечислим некоторые из задач логического вывода, с которыми вы можете столкнуться при использовании моделей НММ.

• Определение наиболее вероятного описания состояний, производящих ряд наблюдений. Включает в себя оценку вероятностей эмиссии этих состояний, а также матрицы переходов, которая показывает вероятность перехода процесса из одного состояния в другое. Мы проделали это выше, хотя и не задавали вероятности перехода в явном виде.

• Определение наиболее вероятной последовательности состояний с учетом ряда наблюдений и описания состояний, а также вероятностей их эмиссии и переходов. Мы также выполнили эту задачу в предыдущем упражнении. Иногда ее называют “наиболее вероятное объяснение”, и для ее решения обычно применяется алгоритм Витерби.

• Фильтрация и сглаживание. В этой ситуации фильтрация будет соответствовать оценке скрытого состояния последнего временного шага с учетом последнего наблюдения. Сглаживание будет соответствовать определению наиболее вероятного распределения скрытого состояния на конкретном временном шаге с учетом наблюдений до, во время и после этого временного шага.

***Байесовский структурный временной ряд***

Байесовский структурный временной ряд (Bayesian structural time series — BSTS) связан с линейной гауссовой моделью, которую мы ранее использовали в фильтрах Калмана. Основное различие состоит в том, что байесовские структурные временные ряды позволяют использовать уже существующие компоненты для построения более сложных моделей, которые отражают известные факты или интересные гипотезы о системе. С их помощью можно разработать структуру модели, провести обучение по имеющимся данным для оценки параметров модели и посмотреть, насколько хорошо модель описывает и предсказывает поведение системы.

Модель BSTS базируется на более сложных математических принципах, чем те, которые использовались в линейной гауссовой модели, которая рассматривалась при изучении фильтра Калмана. Ниже приведен только краткий их обзор, а также пример реализации в программном коде.

Обучение модели BSTS выполняется в четыре этапа, следующих в таком порядке.

1. Определение структурной модели, в частности задание априорных вероятностей.

2. Применение фильтра Калмана для обновления оценок состояния на основе наблюдаемых данных.

3. Применение метода “пик-плато” для выбора переменных в структурной модели (познакомиться с методом “пик-плато” можно в Википедии. Его математика довольно сложная, и мы не будем останавливаться на ее детальном рассмотрении. Метод “пик-плато” наиболее востребован в системах с большим количеством входных данных, которые нужно описать упрощенной моделью с небольшим количеством переменных).

4. Усреднение по байесовской модели для объединения результатов с целью составления прогноза.

В следующем примере мы сосредоточимся только на шагах 1 и 2, в которых определяется гибкость модели, основанной на существующих модульных компонентах, а затем обучим ее на имеющихся данных с помощью байесовского метода, обновляющего оценку параметров с течением времени.

***Код реализации байесовских структурных временных рядов***

В дальнейшей работе будем использовать популярный и невероятно производительный пакет bsts, разработанный Google, и открытый набор данных, полученный с ресурса OpenEI.org.

Отобразим исходные данные на графике, чтобы получить представление о том, что именно нам необходимо смоделировать.

## R

elec = f read ("electric, csv")

require(bsts)

n = colnames (elec) [9]

par(mfrow = c(2, 1))

plot(elec[ [n] ] [1:2000])

plot (elec [ [n] ] [1:96)

## Как указывалось ранее, для правильного анализа временных рядов

## важно правильно подобрать временную шкалу

Просмотр данных позволяет определиться с принципами моделирования. На графике достаточно четко просматривается дневной шаблон, и существует вероятность получения подобного шаблона для недельного изменения данных. Такие шаблоны отражают сезонное поведение данных, которое нужно отразить в модели. Кроме того, в ней нужно учесть присутствующий в данных тренд, представляющий нестационарное поведение, наблюдаемое в верхней области построения.

## R

ss <- AddLocalLinearTrend(list (), elec[[n]])

ss <- AddSeasonal(ss, elec[[nJ], nseasons = 24, season.duration = 1)

ss <- AddSeasonal (ss, elec[[nJ], nseasons = 7, season.duration = 24)

Локальный линейный тренд модели отображает тот факт, что среднее значение и наклон линии тренда описываются случайным блужданием (дополнительная информация приведена на сайте https://perma.cc/2N77-ALJ4).

Сезонная составляющая модели представлена двумя аргументами, один из которых указывает количество сезонов, а другой — продолжительность сезона. Первая добавляемая в модель сезонная составляющая определяет дневной цикл. В нее нужно включать сезоны почасового изменения данных — каждый длительностью один час. Вторая сезонная составляющая представляет недельный цикл. Нам нужно добавить в нее сезоны для каждого дня недели, каждый из которых длится 24 часа.

Вы можете задаться вопросом, действительно ли нужно начинать наблюдения в 12:01 в понедельник (или определить неделю другим способом). В данном случае согласованность данных важнее, чем назначение сезонной подписи первому дню недели. На самом деле в наблюдаемой повторяющейся структуре анализ сезонности можно выполнять при абсолютно любом способе разбивки данных на 24 часовые интервалы.

Ниже приведена наиболее сложная в вычислительном отношении часть кода. Преимущество пакета bsts заключается в возможности проведения вычислений по методу Монте-Карло с использованием марковских цепей (МСМС — Markov Chain Monte Carlo).

## R

modell <- bsts(elec[[n] ],

state.specification = ss,

niter = 100)

plot(modell, xlim = c(1800, 1900)

Всегда можно проверить сезонные компоненты. Например, сезонная составляющая для дней недели проявляется следующим образом.

## R

plot(modell, "seasonal", nseasons = 7, season.duration = 24)

Недельная сезонная составляющая демонстрирует хорошую устойчивость, в то время как дневная сезонность, показанная на графике, сильно зависит от времени суток (вероятнее всего, привязана к световому дню).

Также проявляется локальный линейный тренд, показывающий общее снижение потребления электроэнергии.

Наконец, построим прогноз апостериорного распределения, отображаемого в графическом виде. Обратите внимание, что модель настолько гибка, что позволяет указать количество временных горизонтов, подлежащих предсказанию.

Помните, что временной горизонт указывается на почасовой основе — прогнозирование вперед на 24 временных горизонта может показаться амбициозным шагом, но на самом деле будет выполняться всего на один день. В нашем примере прогнозирование проводится для 72 периодов времени, что определяется контекстом задачи.

pred <- predict(model1, horizon = 24, quantiles = c(0.05, 0.95))

plot(pred, plot.original = 72)

В пакете bsts и байесовском моделировании структурных временных рядов есть немало возможностей, которые подлежат отдельной параметризации.

• Возможность указывать нестандартные априорные распределения.

• Выбор регрессоров с помощью метода "пик-плато".

• Усреднение байесовской модели. Всю необходимую информацию о реализации этих возможностей средствами пакета bsts вы найдете в его документации.

Вы получили только поверхностное представление о задачах, возлагаемых на модели BSTS. Ниже перечислено несколько важных особенностей, характеризующих сильные стороны технологии BSTS. • Модели BSTS позволяют выполнять моделировании с учетом любых априорных распределений. Стандартная линейная гауссова модель, которую мы рассмотрели при обсуждении фильтра Калмана, является лишь одним из вариантов классического априорного распределения, в то время как модель BSTS работает с множеством других вариантов (например, асимметричными априорными вероятностями).

• В BSTS разрешается самостоятельно выбирать переменные.

• Модели BSTS можно комбинировать с методами усреднения байесовских моделей, что позволяет устранить неопределенность, связанную с выбором модели.

Хотя в текущем примере данные возможности не учитывались, все они доступны для реализации с помощью пакета bsts. Вы можете найти много соответствующих примеров в Интернете.

***Дополнительные источники***

• Фильтры Калмана и линейные гауссовы модели пространства состояний Greg Welch and Gary Bishop, “An Introduction to the Kalman Filter” technical report, University of North Carolina at Chapel Hill, 1995, https: //perma.cc/ZCU8-MXEF

Вводный курс по фильтру Калмана, содержащий базовые сведения и описание принципов матричных вычислений, применяемых при его расчете. Включает рассмотрение расширенного фильтра Калмана и наиболее распространенных сценариев его практического применения — описания нелинейных процессов или учета нелинейных ошибок измерения.

R.E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,”Transactions of the ASME—Journal of Basic Engineering 82, Series D (1960): 35- 45, https://perma.cc/GNC4-YLECСтатья 1960 года, включающая исходное описание фильтра Калмана.

Для ее понимания достаточно базовых познаний в статистике и математическом анализе. Представляет интерес с исторической точки зрения, поскольку содержит упоминание об исходном назначении фильтра Калмана и мотивационном контексте его создания.

R. Labbe, “Kalman and Bayesian Filters in Python,” GitHub repository, https: //perma.cc/CMU5-Y94A

Репозиторий GitHub, содержащий десятки примеров использования фильтров Калмана и связанных с ними общих методов фильтрации. Структурирован как учебник: включает рабочие упражнения, книгу в формате PDF и сборник заданий с примерами их решения.

Marie Auger-Methe et al., “State-Space Models' Dirty Little Secrets: Even Simple Linear Gaussian Models Can Have Estimation Problems,” Scientific Reports 6, no. 26677 (2016), https://perma.cc/9D8V-Z7KJ

В этой статье освещается случай подверженности простых линейных гауссовых моделей, подобных применяемым при изучении фильтра Калмана, заведомо ошибочной спецификации, особенно в случаях относительно больших ошибок измерения значений временного ряда. В материале статьи акцент делается на решении экологических задач, но рассматриваемая в ней проблематика будет близкой для множества других дисциплин, основанных на управлении данными. В ней предлагаются совершенно иные подходы, обладающие собственными преимуществами, по сравнению с принятыми в исходном методе.

• Скрытые марковские модели

Andrew Moore, “Hidden Markov Models,” lecture notes, School of Computer Science, Carnegie Mellon University, https://perma.cc/K3HP-28T8

Лекционные материалы с обзорным описанием методов НММ, включающие иллюстрации алгоритмов оценки и примеры практического применения НММ в робототехнических приложениях.

Dan Klein, “Artificial Intelligence: Hidden Markov Model,” lecture notes, University of California Berkeley, https: //perma.cc/V7U4-WPUA

Еще один справочник по методам НММ. В нем описана роль НММ в технологиях оцифровки речи, а также алгоритмах искусственного интеллекта в стратегических играх.

user34790, “What Is The Difference Between the Forward-Backward and Viterbi Algorithms?” question posted on Cross Validated, StackExchange, July 6, 2012, https://perma.cc/QNZ5-U3CN

Пост на сайте StackExchange, вызвавший активное обсуждение и предлагающий интересные решения, которые находят широкое применение в алгоритмах оценки, используемых в методах НММ. Его материал поможет разобраться в принципах применения НММ для изучения данных временных рядов даже в случаях отсутствия заинтересованности в детальном изучении алгоритмов моделирования.

• Байесовские структурные временные ряды

Mark Steel, “Bayesian Time Series Analysis,” in Macroeconometrics and Time Series Analysis, ed. Steven N Durlauf and Lawrence E. Blume (Basingstoke, UK: Palgrave Macmillan, 2010), 35-45, https: //perma. cc/578D-XCVH

Статья предлагает всестороннее описание байесовских принципов анализа временных рядов и включает краткое обсуждение сильных и слабых сторон рассматриваемых методов.

Steven Scott and Hal Varian, “Predicting the Present with Bayesian Structural Time Series,” unpublished paper, June 28, 2013, https: //perma. cc/4EJX-6WGA

Документ Google, основанный на данных экономических временных рядов и являющийся результатом решения задачи прогнозирования применимо к данным за указанный период времени, которые представляются с различными задержками. В частности, авторами предпринята попытка спрогнозировать уровень безработицы по данным поисковых запросов Google — задачи, в которой сведения о безработице публикуются периодически, а поисковые запросы обрабатываются непрерывно.

Фактически такое “текущее прогнозирование”, несмотря на временные задержки в сборе данных, выполняется для настоящего момента времени. Задача решается с помощью комбинации байесовских структурных временных рядов и ансамблевых методов.

Jennifer Hoeting et al., “Bayesian Model Averaging: A Tutorial,” Statistical Science 14, no. 4 (1999): 382-401, https://perma.cc/BRP8-Y33X

В этой статье приведено детально описание принципов усреднения байесовской модели, находящих применение в самых разных методах. В ней показано, что усреднение байесовской модели позволяет избавиться от неопределенности, вызванной выбором неоптимальной модели. Хорошо проработанные примеры позволяют предельно точно оценить неопределенность в составляемых прогнозах.