

Семинар: Байесовский Анализ Данных

30 января 2026 г.

Содержание

1	Список разобранных на семинаре примеров	3
2	Семинары 10.10.2025 и 24.10.2025	4
2.1	Как обучается классификатор	4
2.2	Реализация NRE	5
2.3	Реализация NPE	7
3	Семинар 07.10.2025	9
4	Практикум Второго Сезона	10
4.1	Практикум А. Изображающие детекторы: деконволюция и реконструкция 2D-треков	10
4.1.1	Проблема 1. Восстановление центроида изображения	10
4.1.2	Проблема 2. Восстановление трека	11
4.2	Задача 2. Работа с гистограммами: Анфолдинг и т.п.	11
4.3	Задача 3. Анализ временных рядов: ИВС и Байес	11
5	Выделение периодических компонент	12
6	Определение числа периодических компонент	16
7	Анфолдинг	17
7.1	Fully Bayesian Unfolding	19
8	Задача Павла Волчугова	26
9	Задача Виктора Кудрявцева	29
9.1	Постановка задачи	31
9.1.1	Функция правдоподобия	32
9.1.2	Прайоры	32
9.1.3	ВАЖНЫЕ ВОПРОСЫ	33
9.2	Фильтры Калмана	33
9.3	Time Series in PyMC	34
10	Задача Романа Сараева	35
10.1	Общая постановка задачи	36
10.2	Байесовская модель стерео-метеора	40

11 Введение в Эльфолокацию	42
11.1 Эльфы «УФ атмосферы»	42
11.2 Простейшая кинематическая модель эльфа	43
11.3 Величина сигнала в детекторе	45
11.3.1 Учет нетривиальной PSF	47
11.4 Двумерная модель с наклонным разрядом	47
11.4.1 Зависимость от функции тока	48
11.5 Трехмерная динамическая модель эльфа	52
11.6 Приложение: генерация данных	52
11.6.1 Трехмерная генерационная модель	54
12 Может ли лучшая модель привести к более широкому HDI?	55
13 Байесовский вывод в SSM	55
A Введение в PyMC	57
A.1 Распределения PyMC	57
A.2 Объект InferenceData и как с ним работать	58
A.3 Sequential MC в PyMC	60
A.4 Гауссовы процессы в PyMC	60
A.4.1 GP: базовый вариант	60
A.4.2 GP: негауссово правдоподобие	62
A.4.3 GP: не-гауссовы процессы и аппроксимации	63
A.4.4 Другие примеры использования GP	63
B STAN vs. PyMC	64
B.1 Полезные примеры STAN	64
C Как выбирать прайор?!	65
C.1 Рекомендации по выбору прайора (по Aki Vehtari)	65
C.1.1 Независимость	65
C.1.2 Общие принципы	65
C.1.3 Насколько информативен прайор?	66
C.1.4 Масштабирование прайоров по умолчанию в зависимости от стандартной ошибки оценки эффекта	67

1 Список разобранных на семинаре примеров

Ниже приводится список тех примеров с сайта разработчиков PyMC, из книг по вероятностному программированию (со ссылками на коды), которые с той или иной степенью детализации разбирались на семинарах БАД в 2024-25 гг.

1. (2024) [Bayesian regression with truncated or censored data](#)
2. (2025) [Model building and expansion for golf putting](#)
3. (04.04.2025) [SMC-ABC and Lotka–Volterra](#)
4. (04.04.2025) [Updating priors](#)

2 Семинары 10.10.2025 и 24.10.2025

Разбирая на семинаре [Tutorial по SBI](#), натолкнулись на проблему того, что не совсем понимаем, как действует алгоритм NRE. Ниже предпринята попытка прояснить ситуацию.

2.1 Как обучается классификатор

Давайте на время отвлечемся от реализации NRE и просто посмотрим на нейросетевой классификатор, перед которым поставлена цель различать два типа данных – условно картинок котиков и зайчиков. Для определенности, пусть котики распределены как $p_1(z)$, а зайчики – как $p_2(z)$. Надо разобраться к чему асимптотически стремится нейросеть, когда обучается с функцией потерь BCE (binary cross entropy).

Итак, пусть последний слой нейросети не имеет активационной функции, т.е. выдает число $l(z) \in (-\infty, +\infty)$ (от Linear), которое для задания бинарной вероятности используется нами в виде $p(y = 0|z) = \sigma(l(z)) \equiv e^l / (1 + e^l) \in [0, 1]$, $p(y = 1|z) = 1 - p(y = 0|z)$. Здесь $y = 0$ и $y = 1$ метки классов, соответственно котика и зайчика.

По определению функция потерь BCE есть $L = -1 \cdot \ln p(y = 0|z)$, если на входе котик, и $L = -1 \cdot \ln p(y = 1|z)$, если это был зайчик¹. Но ведь котики и зайчики к нам поступают в соответствии с распределениями $p_0(z)$ и $p_1(z)$, поэтому средняя функция потерь равна

$$L[d(z)] = - \int dz \{ p_0(z) \ln d(z) - p_1(z) \ln(1 - d(z)) \}$$

где мы для упрощения записи обозначили $d(z) \equiv \sigma(l(z)) = p(y = 0|z)$.

Примечание. Важно не путать вероятности $p(y = 0|z)$ и $p_{0,1}(z)$. Первая фактически просто выход нашей нейросети (после взятия сигма-функции), а две оставшиеся – вероятности подачи на вход нейросети обучающих картинок соответствующего класса (кстати, а откуда они берутся?).

Теперь давайте посмотрим какая функция $d^*(z)$ минимизирует функционал потерь. Для этого воспользуемся функциональной производной, $\frac{\delta L[d(z)]}{\delta d(z)} = \frac{p_0(z)}{d(z)} - \frac{p_1(z)}{1-d(z)}$, приравняв которую нулю, получаем

$$d^*(z) = \frac{p_0(z)}{p_0(z) + p_1(z)}$$

Другими словами, если обучать нейросеть картинками с распределениями $p_0(z)$ и $p_1(z)$ а в качестве функции потерь взять BCE, сигма-функция от (линейного) выхода будет асимптотически (после длительного обучения) аппроксимировать $d^*(z)$.

Теперь вернемся к примеру из руководства. Там в качестве z фигурировала совокупность (x, θ) (данные и параметр), а два класса в обучающей выборке создавались так, чтобы

$$p_0(z) = p(x, \theta), \quad p_1(z) = p(x) p(\theta)$$

Напомню, что первое распределение создавалось с помощью сэмплирования θ из прайора $p(\theta)$ с последующим генерацией соответствующего x из стохастического симулятора. А второе распределение собиралось из них же, но предварительно перемешав номера θ (тем самым снималась корреляция между θ и x).

Тогда

$$d^*(x, \theta) = \frac{p(x, \theta)}{p(x, \theta) + p(x) p(\theta)} = \frac{p(x|\theta) p(\theta)}{p(x|\theta) p(\theta) + p(x) p(\theta)} = \frac{p(x|\theta)}{p(x|\theta) + p(x)} = \frac{1}{1 + p(x)/p(x|\theta)}$$

¹То есть она минимизируется если я поданного на вход котика с большой вероятностью называю котиком и аналогично с зайчиком.

А так как $d(x, \theta) \equiv \sigma(l(x, \theta))$, то выход нейросети $l(x, \theta)$ в результате обучения стремится к

$$l^*(x, \theta) = \ln \frac{p(x|\theta)}{p(x)}$$

т.е. с точностью до аддитивной константы (не зависящей от θ) становится равным логарифму правдоподобия. Именно это и используется в дальнейшем для формирования процесса МСМС сэмплирования из постериорного распределения (к логарифму правдоподобия добавляется логарифм прайора).

2.2 Реализация NRE

При подготовке к семинару Роман основательно поработал над ноутбуком тьюториала. В первую очередь, это коснулось преобразованию нейросетевых SBI с Torch в Keras. Это позволило реализовать NRE без использования функционала классов и сделала код максимально прозрачным. Приведем соответствующий кусочек кода (полный вариант ноутбука доступен здесь: ???).

Примечание. Для удобства восприятия Роман переобозначил данные: теперь предикаторные переменные (те, что по горизонтальной оси графика данных) обозначены как x , а сами отсчеты (по вертикальной оси) – как y . Так что в качестве логит-выхода нейросети фигурирует $l^*(y, x, \theta) = \ln \frac{p(y|x, \theta)}{p(y|x)}$. Считается, что предикаторные переменные фиксированы (т.е. совпадают с теми, что использованы в измерениях): $x = \text{observed_x}$.

Сначала подготовим данные для обучения нейросетевого классификатора (bump_simulator – стохастический симулятор данных): *SSh: Что такое tqdm()?*

```
n_train = 100_000
gen = np.random.default_rng(1337)
theta_samples = gen.uniform(low=[0, 0], high=[200, 1], size=(n_train, 2))
y_samples = np.array([bump_simulator(theta, observed_x, gen) for theta in tqdm(theta_sam
y_mean = y_samples.mean(axis=0)
y_std = y_samples.std(axis=0)
y_samples = (y_samples - y_mean) / y_std
theta_mean = theta_samples.mean(axis=0)
theta_std = theta_samples.std(axis=0)
theta_samples = (theta_samples - theta_mean) / theta_std

# H0
theta_normal = np.array(theta_samples)
inputs_H0 = np.concatenate([theta_normal, y_samples], axis=1)
outputs_H0 = np.ones(n_train, dtype=float)
# H1
theta_shuffled = np.array(theta_samples)
gen.shuffle(theta_shuffled)
inputs_H1 = np.concatenate([theta_shuffled, y_samples], axis=1)
outputs_H1 = np.zeros(n_train, dtype=float)

train_x = np.concatenate([inputs_H0, inputs_H1])
train_y = np.concatenate([outputs_H0, outputs_H1])
```

Затем определим саму нейросеть из 5-ти полносвязных слоев (с функцией активации gelu) и скомпилируем ее

```

import tensorflow.keras as keras
X_DIM, THETA_DIM = 50, 2
INPUT_DIM = X_DIM+THETA_DIM
neural_ratio_estimator = keras.Sequential([
    keras.layers.Input(shape=(INPUT_DIM,)),
    keras.layers.Dense(128, activation="gelu", name="input_to_hidden"),
    keras.layers.Dense(128, activation="gelu", name="hidden_1"),
    keras.layers.Dense(128, activation="gelu", name="hidden_2"),
    keras.layers.Dense(128, activation="gelu", name="hidden_3"),
    keras.layers.Dense(128, activation="gelu", name="hidden_4"),
    keras.layers.Dense(1, activation="linear", name="output"),
])
neural_ratio_estimator.compile(
    optimizer = keras.optimizers.Adam(3e-4),
    loss = keras.losses.BinaryCrossentropy(from_logits=True),
    metrics = ["binary_accuracy"]
)
neural_ratio_estimator.summary()

```

Обращаем внимание, что на выходе сети один-единственный линейный узел – это и есть логит нашего классификатора. (При задании функции потерь в виде ВСЕ это надо указать отдельно с помощью аргумента `from_logits`.)

Общее число параметров сети велико – 72 961, но обучается она довольно быстро (Роман фиксирует число эпох – 20):

```
neural_ratio_estimator.fit(train_x, train_y, batch_size=128, epochs=20)
```

После этого все готово для использования обученной сети в байесовском выводе на основе МСМС. Только надо не забыть нормализовать данные – ведь нейросеть мы обучали именно при таких входах...

```

def log_like(theta, y):
    """ Log-likelihood ratio estimator using trained classifier logits.
    """
    y = (y - y_mean) / y_std
    theta = (theta - theta_mean) / theta_std
    cat = np.concatenate([theta, y])
    cat = np.expand_dims(cat,0)
    #return neural_ratio_estimator(cat, training=False).numpy().squeeze()
    return neural_ratio_estimator.predict_on_batch(cat).squeeze()

def log_post(theta, y):
    """ Log-posterior distribution, for sampling.
    """
    lp = log_prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    else:
        return lp + log_like(theta, y)

```

Conditional posterior density estimation

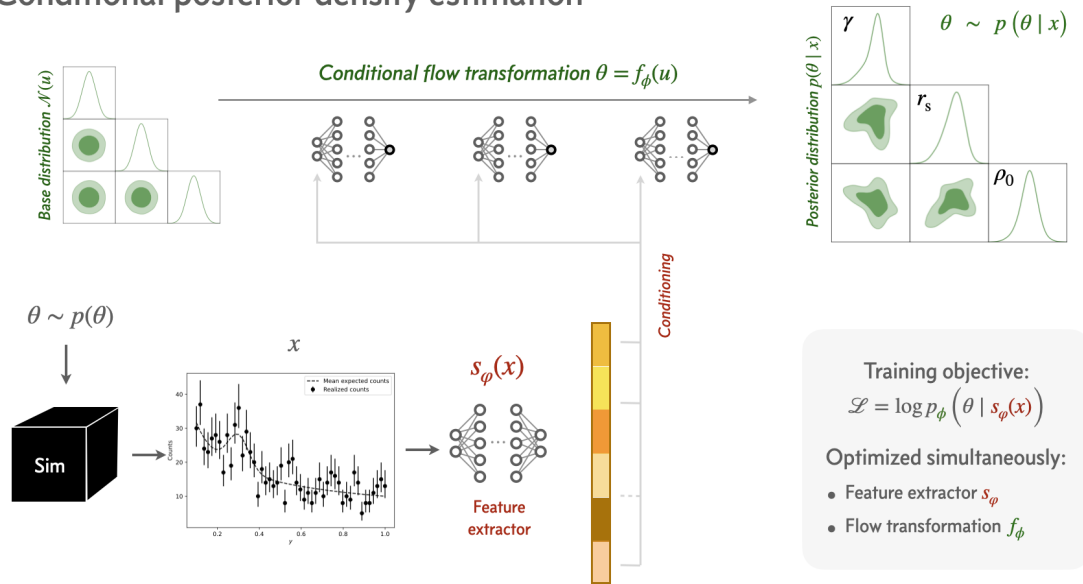


Рис. 1: Схематическое представление алгоритма NPE.

`ndim, nwalkers = 2, 32`

`sampler = emcee.EnsembleSampler(nwalkers, ndim, log_post, args=(observed_y,))`

`pos = opt.x + 1e-3 * np.random.randn(nwalkers, ndim)`

`sampler.run_mcmc(pos, 5000, progress=True);`

Результат показывает, что качество аппроксимации постериорного распределения заметно выросло по сравнению с ABC (это легко установить, так как в этой простой задаче у нас есть «золотой стандарт» – МСМС с явно заданным правдоподобием). Однако все равно присутствует заметное смещение и уширение – особенно в маргинале $p(\mu_s | \mathcal{D})$.

2.3 Реализация NPE

Получить более точную аппроксимацию $p(\mu_s, A_s | \mathcal{D})$ можно при помощи метода NPE.

Примечание. Используемая в tutorialе реализация NPE базируется на библиотеке [nflows](#) – а comprehensive collection of normalizing flows using PyTorch. Нормализующие потоки детально рассматриваются в цикле работ George Papamakarios, с которыми мы рекомендуем ознакомиться по его [PhD-диссертации](#) (так как в ней они приведены в исправленном виде). В одной из последующих своих работ Papamakarios et al. пытается соединить NRE и NPE в рамках единого подхода, см. [\[Durkan2020\]](#).

Архитектуру нейросети оставим той же самой: *SSH: Чем отличается tfk от keras?*

```
import tensorflow_probability as tfp
tfd = tfp.distributions
tfb = tfp.bijectors
import tf_keras as tfk
```

```
context_len = 16
```

```
featurizer = tfk.Sequential([
```

```

tfk.layers.Input(shape=(X_DIM,)),
tfk.layers.Dense(128, activation="gelu", name="input_to_hidden"),
tfk.layers.Dense(128, activation="gelu", name="hidden_1"),
tfk.layers.Dense(128, activation="gelu", name="hidden_2"),
tfk.layers.Dense(128, activation="gelu", name="hidden_3"),
tfk.layers.Dense(128, activation="gelu", name="hidden_4"),
tfk.layers.Dense(context_len, activation="linear", name="output"),
])
base_dist = tfd.Sample(tfd.Normal(loc=0., scale=1.), sample_shape=[2])

```

Напомним, что в NPE-подходе цель нейросети – совершить переход (трансформацию) от исходных параметров θ к параметрам, в которых постериорное распределение станет многомерным нормальным. При этом используется MAF – masked autoregressive flow, который..., см. рис. 1

SSh: НАДО ДАТЬ РАЗВЕРНУТЫЕ ПОЯСНЕНИЯ!!!

```

n_layers = 4
transforms = []
for i in range(n_layers):
    transforms.append(tfb.MaskedAutoregressiveFlow(
        tfb.AutoregressiveNetwork(params=2, hidden_units=[32],
                                   event_shape = (THETA_DIM,),
                                   conditional=True,
                                   conditional_event_shape=(context_len,)),
        name=f"maf_{i}",
    ))
    print(transforms[-1].name)
transform = tfb.Chain(list(reversed(transforms)))
transformed_dist = tfd.TransformedDistribution(base_dist, transform)

theta_ = tfk.layers.Input(shape=(THETA_DIM,), dtype=tf.float32)
y_ = tfk.layers.Input(shape=(X_DIM,), dtype=tf.float32)
c_ = featurizer(y_)
kwargs_for_all_steps = dict(conditional_input=c_)

log_prob_ = transformed_dist.log_prob(theta_, bijector_kwargs = {
    "maf_0":kwargs_for_all_steps,
    "maf_1":kwargs_for_all_steps,
    "maf_2":kwargs_for_all_steps,
    "maf_3":kwargs_for_all_steps,
})

model = tfk.Model([theta_, y_], log_prob_)
model.compile(optimizer=tfk.optimizers.Adam(learning_rate=3e-4),
              loss=lambda _, log_prob: -log_prob)

Обучение проходит чуть медленнее

model.fit(x=[theta_samples, y_samples], y=np.zeros((n_train, 0), dtype=np.float32),
         batch_size=128, epochs=20,
         shuffle=True, verbose=True)

```

```
features = featurizer.predict_on_batch(np.expand_dims((observed_y-y_mean)/y_std,0))

SAMPLES = 50_000
common_kwargs = {'conditional_input': np.repeat(features, SAMPLES, axis=0)}
bijector_kwargs = {
    "maf_0":common_kwargs, "maf_1":common_kwargs,
    "maf_2":common_kwargs, "maf_3":common_kwargs,
}

result_npe = transformed_dist.sample(
    (SAMPLES,),
    bijector_kwargs=bijector_kwargs).numpy().squeeze()*theta_std+theta_mean
```

зато качество аппроксимации удивительное – практически идентично «золотому стандарту»!

3 Семинар 07.10.2025

На этом семинаре, помимо прочего, была предпринята попытка разобраться в заключительном примере [SBI-Tutorial](#) – анализу размытого двумерного изображения набора точечных источников.

4 Практикум Второго Сезона

4.1 Практикум А. Изображающие детекторы: деконволюция и реконструкция 2D-треков

Одним из распространённых видов детектирования является построение изображения на фоточувствительной поверхности с помощью специально организованной оптической системы. В простейшем случае такой детектор может создавать одно интегральное изображение, накапливая сигнал в течение какого-то интервала времени. Но для нас наиболее интересными будут так называемые *детекторы динамических изображений*, которые формируют последовательность снимков с запечатленными на них различными стадиями развития того или иного физического явления. Последнее может меняться со временем – перемещаться или трансформироваться в пространстве, варьироваться по интенсивности излучения, исчезать и появляться. Такие быстроменяющиеся процессы мы будем называть *транзиентными* (или просто транзиентами).

4.1.1 Проблема 1. Восстановление центроида изображения

В этой задаче, если не оговорено особо, детектор будет представлять собой матрицу из квадратных сенсоров (пикселей), плотно расположенных на фокальной плоскости (ФП) оптической системы. Нас будет интересовать отклик детектора на удаленный точечный источник, который превращается оптической системой в «пятно» на ФП. Чтобы привязаться к какой-то из точек такого пятна, введем понятие *центроида* пятна \mathbf{R}_0 , т.е. «центр масс» изображения при его регистрации детектором с бесконечно малыми пикселями.

Целью задачи является нахождение \mathbf{R}_0 по одному-единственному изображению, построенному на ФП с пикселями конечных размеров (будем считать их размер единичным). Конечно, изображение зашумлено. Для простоты среднее значение уровня фона будем считать известным (и приравняем его 0), поэтому величину шума можно характеризовать стандартным отклонением фоновых флуктуаций (кстати, не обязательно гауссовых!).

Во всех изложенных ниже задачах требуется реконструировать положение центра изображения байесовским методом и сравнить с более «классическими» подходами: *Varycenter*, *MLE*, *MAP*. В продвинутом варианте этой проблемы (задача со «звездочкой») предлагается реализовать один или несколько вариантов *SBI*-реконструкции.

Задача 1а. «Гауссово изображение»

Дано: $\mathbf{R}_0(X_0, Y_0)$, σ_{psf} , E_0

Сгенерить: $\mathcal{D} = \{d_i\}$

Найти: реконструированное положение центра \mathbf{R} .

Пояснение: Гауссово изображение характеризуется своим центром, совпадающим с центроидом), и характерной шириной, в качестве которой мы выберем параметр σ_{psf} (все линейные размеры в единицах размера пикселя). Качество реконструкции в общем случае может зависеть от смещения центроида из центра пикселя, а кроме того, от отношения сигнала к шуму. Чтобы учесть этот последний эффект будем выражать интегральное значение энергии пятна E_0 в единицах шума в одном пикселе.

Примечание. Обращаем внимание, что сигнал в одном пикселе меньше E_0 и зависит от \mathbf{R}_0 . Приведем такую оценку: если центроид совпадает с центром пикселя, то энергия в пикселе заключена между $E_0(1 - \exp(-1/8\sigma_{\text{psf}}^2))$ и $E_0(1 - \exp(-1/4\sigma_{\text{psf}}^2))$ (соответственно вписанных и описанных в квадрат кружок пятна). *SSH: Приведите точную оценку!*

Результатом байесовской реконструкции является постериорное распределение $p(X, Y|\mathcal{D})$. Так как истинное положение центроида нам известно, то точность реконструкции можно

охарактеризовать величиной смещения $r = |\mathbf{R}_m - \mathbf{R}_0|$ и шириной постериорного распределения. В качестве центра распределения \mathbf{R}_m может выступать, например, его мода, среднее или медиана (проверить все варианты), а меру ширины предлагаем выбрать в виде стандартных отклонений по обеим осям ФП, σ_X и σ_Y .

Исследовательская часть задачи заключается в выяснении зависимости так понимаемой точности реконструкции от основных входных параметров – $X_0, Y_0, \sigma_{\text{psf}}, E_0$. В качестве дополнительной задачи можно посмотреть как эти результаты поменяются при неточной спецификации интерпретирующей модели в ситуации с негауссовым шумом.

1b. «Краевые эффекты»

Решите предыдущую задачу, рассмотрев ФП не полностью заполненную пикселями (т.е. с границами поля зрения). Самый простой способ – проведите реконструкцию центроида при его приближении к краю или к углу поля зрения.

Вторым вариантом деформации задачи в сторону ее большей реалистичности попробуйте учесть неточность знаний о коэффициентах чувствительности детектора s_i . Для этого считайте их нормально-распределенными со средним 1 и некоторым (единым для всех пикселей) стандартным отклонением σ_s . Сравните случай когда σ_s вам известно (например, получено в отдельном высокоточном калибровочном эксперименте) и когда ее величину надо оценить в рамках реконструкции. А насколько испортит реконструкцию существенно неравномерное распределение чувствительностей?

1с. «Не-гауссовы изображения»

Наиболее интересным способом мис-спецификации является интерпретация изображения как гауссова, тогда как на самом деле оно сгенерировано по какому-то иному распределению. В качестве вариантов могут быть использованы Mofat (посредством Numerical Integration), Сома (МС) или их комбинации. Интересным способом размытия идеального изображения будет использование конволюции с заранее выбранным двумерным фильтром – только задавать этот фильтр придется на более детализированной матрице субпикселей. (При этом появляется широкий спектр возможностей — начинаю от направленного «заблериования» и заканчивая...)

Во всех этих случаях сначала осуществите реконструкцию с гауссовой моделью изображения и посмотрите насколько ухудшились ваши предсказания. После этого с помощью SBI можете попробовать реализовать более тонкую модель, например заданную фильтром специального вида (симулятор таких данных весьма прост и быстр).

4.1.2 Проблема 2. Восстановление трека

2а. «Постоянный сигнал»

2b. «Разорванный трек»

2с. «Плавно меняющийся сигнал»

4.2 Задача 2. Работа с гистограммами: Анфолдинг и т.п.

4.3 Задача 3. Анализ временных рядов: ИВС и Байес

5 Выделение периодических компонент

Демонстрация байесовского подхода на задаче выделения в сигнале периодических компонент хороша тем, что ее можно проводить как в режиме монте-карловского сэмплирования (т.е. средствами вероятностного программирования), так и аналитически. Формулировка байесовской модели в РумС в случае с заранее известным числом компонент не представляет труда с аналитической точки зрения, поэтому здесь мы рассмотрим именно этот подход. Подробно он изложен в документе Modern Data Analysis (далее просто MDA, см. там раздел 3.1) и в диссертации и работах Бретхортса (см. работу G.Larry Bretthorst, Bayesian Spectrum analysis and Parameter Estimation).

Пусть даны N эквидистантных измерений d_k , в которых мы пытаемся обнаружить следы некоторой *периодической функции*, т.е. $d_k = g(t_k) + \xi_k$, причем $g(t + 1/f) = g(t)$ с неизвестной нам частотой f и независимыми ошибками измерения ξ_k .

Ограничимся случаем эквидистантной выборки $\{t_k\}$, тогда общее время измерений $T = (N - 1)\tau$, где τ – интервал дискретизации (по времени). Для простоты также будем считать, что на стадии предобработки из сигнала было вычтено средневывборочное значение, поэтому $\bar{d} \equiv \frac{1}{N} \sum_k d_k = 0$.

Существует большое разнообразие «классических» фреквентистских методов распознавания в зашумленном сигнале периодической компоненты, большинство из них опирается на понятие периодограммы.

Периодограммой Шустера называют функцию²

$$\mathcal{P}(\omega) = \frac{1}{N} \left| \sum_{k=1}^N d_k e^{i\omega t_k} \right|^2 = \frac{1}{N} (\mathcal{R}^2 + \mathcal{I}^2)$$

где

$$\mathcal{R} = \sum_{k=1}^N d_k \cos(\omega t_k), \quad \mathcal{I} = \sum_{k=1}^N d_k \sin(\omega t_k) \quad (1)$$

представляют собой вещественную и мнимую часть преобразования Фурье измеренного сигнала.

Периодограмма является функцией непрерывной (циклической) частоты ω , но определенный дискретный набор частот имеет особый статус. Речь идет от так называемых *фурье-частот* $\omega_n = n \cdot 2\pi/T$. С одной стороны, это именно те частоты, для которых «работает» быстрое преобразование Фурье (FFT): $F[d]_n = \sum_k d_k \exp\{-i2\pi nk/(N - 1)\}$. С другой, можно показать, что для белого шума, $d_k = \xi_k$, $\mathcal{P}(\omega_n)$ оказываются независимо распределенными и каждое имеет распределение χ_2^2 – распределение хи-квадрат с двумя степенями свободы, эквивалентное экспоненциальному распределению (со средним 2)³. На этих свойствах *независимости* и *экспоненциальности распределения* и основаны многие фреквентистские рецепты, эксплуатирующие понятие *p-value*, т.е. отвержение нулевой гипотезы (об отсутствии периодической компоненты) на определенном уровне значимости⁴.

Наконец, периодограмма фигурирует и при подгонке по методу наименьших квадратов (МНК) сигнала параметрически заданной гармоникой $A \cos(\omega t + \phi)$: если при минимизации квадратичной функции ошибок подставить в нее оптимальные оценки для \hat{A} и $\hat{\phi}$,

²В разных статьях и учебниках можно встретиться с различной нормировкой, так, в фигурирующей ниже в тексте достаточной статистике C множитель $/1N$ отсутствует.

³Конечно, в данном случае речь идет о предварительно стандартизованном сигнале, т.е. сигнале с $\bar{d} = 0$ и $\bar{d}^2 = 1$.

⁴Обращаем внимание, что на других частотах, $\omega \neq \omega_n$, это свойство отсутствует и для отвержения нулевой гипотезы приходится пользоваться методами Монте-Карло.

то полученная таким образом *редуцированная ошибка* будет с точностью до константы и знака совпадать с $\mathcal{P}(\omega)$. Другими словами, в рамках вероятностной интерпретации МНК, когда речь идет о максимизации функции правдоподобия $\mathcal{L}(\omega, A, \phi)$, получается *профильное правдоподобие* $\mathcal{L}_p(\omega) = \mathcal{L}(\omega, \hat{A}(\omega), \hat{\phi}(\omega)) = \text{const} + \mathcal{P}(\omega)/\sigma^2$, где σ^2 – дисперсия ошибок измерения.

Задание 1. Вычислите периодограмму Шустера для каждого из сгенерированных сигналов, изобразите ее график и проанализируйте его.

Будьте внимательны: в общем случае периодограмма имеет мультимодальный характер и сильно варьируется на масштабах $\sim 1/T$ (напомним, T – общее время измерений).

Периодограмма возникает также в рамках байесовского подхода. Точнее, речь идет об ее обобщении на случай произвольной (не эквидистантной) выборки:

$$C(\omega) = \frac{(1-c)\mathcal{R}^2 + (1+c)\mathcal{I}^2 - 2s\mathcal{R}\mathcal{I}}{1-c^2-s^2}, \quad (2)$$

где были введены обозначения

$$c \equiv \frac{1}{N} \sum_{k=1}^N \cos(2\omega t_k), \quad s \equiv \frac{1}{N} \sum_{k=1}^N \sin(2\omega t_k). \quad (3)$$

Бретхорст показал, что постериорная вероятность для модели с одной гармоникой, параметризованной как $A \cos \omega t + B \sin \omega t$, в случае, когда стандартное отклонение гауссовых ошибок измерения σ задано, имеет вид

$$p(\omega|\mathcal{D}, \sigma) \propto \exp\{C(\omega)/(N\sigma^2)\}. \quad (4)$$

Если же σ не дано, то постериор получается путем маргинализации:

$$p(\omega|\mathcal{D}) = \int d\sigma p(\sigma) p(\omega|\mathcal{D}, \sigma) \propto \left[1 - \frac{2C(\omega)}{N^2 \bar{d}^2}\right]^{-(N-2)/2}. \quad (5)$$

Здесь $\bar{d}^2 = \frac{1}{N} \sum_k d_k^2$ – средневыборочная дисперсия измеренного сигнала (с учетом того, что $\bar{d} = 0$), в обоих случаях предполагались широкие (неинформативные) равномерные прайоры на амплитуды A и B и масштабнo-инвариантный прайор Джеффри⁵ на σ во втором случае, $p(\sigma) \propto 1/\sigma$.

Задание 2. Вычислите функцию $C(\omega)$, постройте ее график и сравните с графиком «классической» периодограммы $\mathcal{P}(\omega)$. Постройте также график (плотности) вероятности $p(\omega|\mathcal{D})$. Проанализируйте результаты.

Задача. Покажите, что в случае эквидистантной выборки сдвиг времени $t \rightarrow t - T/2$ приводит к тому, что $s = 0$ – это значительно упрощает расчеты, так как при этом $C(\omega) = \mathcal{R}^2/(1+c) + \mathcal{I}^2/(1-c)$, а $c = \frac{1}{N} \sin(N\omega\tau)/\sin(\omega\tau)$.

Задача со звездочкой. Покажите, что постериорная оценка дисперсии ошибок измерения (при прайоре Джеффри) при заданной ω

$$\langle \sigma^2 | \mathcal{D}, \omega \rangle = \frac{N\bar{d}^2}{N-4} \left(1 - \frac{2C(\omega)}{N^2 \bar{d}^2}\right)$$

Подсказка: $p(\sigma|\mathcal{D}, \omega) \propto p(\mathcal{D}|\omega, \sigma) p(\sigma)$ и надо воспользоваться интегральным определением гамма-функции, $\Gamma(x) = \int_0^\infty dt e^{-t} t^{x-1}$, и $\Gamma(x+1)/\Gamma(x) = x$.

⁵В прайоре Джеффри (Jeffreys) предполагается, что есть минимальное и максимальное значение параметра, т.е. $p(\sigma) = [\sigma \ln(\sigma_{\max}/\sigma_{\min})]^{-1}$.

А в MDA даже приводится оценка относительной точности дисперсии ошибок:

$$\varepsilon \equiv \sqrt{\langle \sigma^4 \rangle - \langle \sigma^2 \rangle^2} / \langle \sigma^2 \rangle = \sqrt{2/(N-6)}.$$

Задача с двумя звездочками. А сможете ли для своих сигналов представить результат байесовского вывода в виде $\omega = \hat{\omega} \pm \sigma_\omega$?! Подсказка: потребуется разложение в ряд Тейлора.

Интересно, что аналитический байесовский вывод возможен и для ряда более сложных моделей. Так, в случае модели с экспоненциальным затуханием,

$$\exp\{-\gamma t\} [A \cos(\omega t) + B \sin(\omega t)],$$

все, что необходимо сделать, это переопределить значения коэффициентов c и s :

$$c \equiv \frac{1}{N} \sum_{k=1}^N \cos(2\omega t_k) \exp\{-2\gamma t_k\}, \quad s \equiv \frac{1}{N} \sum_{k=1}^N \sin(2\omega t_k) \exp\{-2\gamma t_k\}. \quad (6)$$

Эти коэффициенты, так же как и достаточная статистика, теперь станут функцией двух переменных, $C = C(\omega, \gamma)$, но все остальные выражения останутся без изменений (кроме упрощения $s = 0$ после сдвига $t \rightarrow t - T/2$ для эквидистантной выборки). Аналогичные утверждения справедливы и в случаях гауссового и лоренцового затухания – получите выражение для новых коэффициентов c и s .

Задание 3. Изобразите $C(\omega, \gamma)$ в виде контурной диаграммы. Ухудшается ли качество восстановления частоты (положение $\hat{\omega}$ и его погрешность σ_ω) при использовании для восстановления «неправильной» модели (т.е. не той, что применялась для генерации данных)?

Задача. Вопрос (с подвохом): А как из «двумерной» $C(\omega, \gamma)$ сделать «одномерную» $C(\omega)$? И вообще, что такое $\hat{\omega}$ и σ_ω в этом случае?

Оказывается, байесовский вывод можно провести аналитически даже для моделей с произвольным количеством гармоник M – именно этому и была посвящена диссертация Бретхорта. В общем случае необходимо совершить поворот в пространстве базисных функций, так что все вычисления проще всего проводить в матричной форме. Ниже мы приводим итоговый результат для частного случая двух гармоник (без затухания),

$$C(\omega_1, \omega_2) = \frac{1}{1 - \alpha_+} \left(\frac{\mathcal{R}_1^2}{1 + c_1} + \frac{\mathcal{R}_2^2}{1 + c_2} - \frac{2\alpha_+ \mathcal{R}_1 \mathcal{R}_2}{c_- + c_+} \right) + \frac{1}{1 - \alpha_-} \left(\frac{\mathcal{I}_1^2}{1 - c_1} + \frac{\mathcal{I}_2^2}{1 - c_2} - \frac{2\alpha_- \mathcal{I}_1 \mathcal{I}_2}{c_- - c_+} \right), \quad (7)$$

с коэффициентами $c_a \equiv \frac{1}{N} \sum_k \cos(2\omega_a t_k)$, $a = 1, 2, +, -$, и

$$\omega_\pm = \frac{\omega_1 \pm \omega_2}{2}, \quad \alpha_\pm = \frac{(c_- \pm c_+)^2}{(1 \pm c_1)(1 \pm c_2)}$$

Здесь были введены статистики

$$\mathcal{R}_{1,2} = \sum_k d_k \cos(\omega_{1,2} t_k), \quad \mathcal{I}_{1,2} = \sum_k d_k \sin(\omega_{1,2} t_k) \quad (8)$$

причем подразумевается предварительный сдвиг по времени $t \rightarrow t - T/2$ (переход к «симметризованному» времени с $\bar{t} = 0$). Постериорные вероятности по-прежнему выражаются через достаточную статистику C посредством (4) и (5).

Задание 4. Изобразите на одной плоскости $\omega_1 - \omega_2$ контуры $C(\omega_1, \omega_2)$ и контуры функции $\mathcal{R}_1^2 + \mathcal{I}_1^2 + \mathcal{R}_2^2 + \mathcal{I}_2^2$, в которую она переходит в пределе нулевых c_a . Сильно ли различаются максимумы этих функций? Кстати, а сколько их?

Несложно показать, что $c_a = \eta(\omega_a \tau)$ с $\eta(x) = \frac{1}{N} \sin(Nx) / \sin(x)$. В пределе $N \gg 1$ все коэффициенты c_a ведут себя как $O(1/N)$ и справедлива аппроксимация

$$C(\omega_1, \omega_2) \approx \mathcal{R}_1^2(1 - c_1) + \mathcal{R}_2^2(1 - c_2) - 2\mathcal{R}_1\mathcal{R}_2(c_- + c_+) + \mathcal{I}_1^2(1 + c_1) + \mathcal{I}_2^2(1 + c_2) - 2\mathcal{I}_1\mathcal{I}_2(c_- - c_+),$$

которую также можно представить в виде

$$C(\omega_1, \omega_2) \approx \{\mathcal{R}_1^2 + \mathcal{I}_1^2 + \mathcal{R}_2^2 + \mathcal{I}_2^2\} - c_1\mathcal{R}_1^2 - c_2\mathcal{R}_2^2 + c_1\mathcal{I}_1^2 + c_2\mathcal{I}_2^2 - 2(c_- + c_+)\mathcal{R}_1\mathcal{R}_2 - 2(c_+ - c_-)\mathcal{I}_1\mathcal{I}_2.$$

Основной член, стоящий в фигурных скобках, факторизуется на сумму двух одинаковых функций, а поправочные слагаемые содержат перекрестные члены.

Наиболее интересна ситуация спектрального дублета, когда $\omega_1 \approx \omega_2$. Тогда в пределе больших N в формуле (7) можно пренебречь всеми коэффициентами кроме $c_- = \eta(\Delta\omega\tau/2) \approx \sin(N\pi\Delta\omega/\omega_s)/(N\pi\Delta\omega/\omega_s)$, где $\Delta\omega = \omega_1 - \omega_2$, $\omega_s = 2\pi/\tau$ – частота сэмпирования и

$$C(\omega_1, \omega_2) = \frac{\mathcal{R}_1^2 + \mathcal{I}_1^2 + \mathcal{R}_2^2 + \mathcal{I}_2^2 - 2c_-(\mathcal{R}_1\mathcal{R}_2 + \mathcal{I}_1\mathcal{I}_2)}{1 - c_-^2} \quad (9)$$

Поправка становится существенной в области $|\omega_1 - \omega_2| < \omega_s/(N\pi)$.

Задание 5. Сгенерив несколько сигналов с уменьшающимся расстоянием между линиями $\Delta\omega$ (и оставляя неизменными все остальные параметры), проследите как меняется поведение функции $C(\omega_1, \omega_2)$. Сделайте вывод.

Примечание. Сформулированные выше задания 1-5 могут быть выполнены и средствами PyMC. В этом случае сэмпирование и использование средств ArViz приведет к построению не периодограм или достаточных статистик C , а самих постериорных распределений. Рекомендуем проделать это и убедиться в идентичности получаемых результатов (на качественном уровне). Примеры программ на Python (и PyMC) для генерации сигнала и его реконструкции приведены ниже. Прайоры на начальные фазы выбраны в классе распределений [VonMises](#) (периодичном на $[-\pi, \pi]$), а прайор Джеффри на σ задан посредством равномерного распределения на $\ln \sigma$. Обращаем также внимание, что в модели двух гармоник, чтобы «отодвинуть» сэмплер от значений периодов вблизи нуля, пришлось ввести фиктивные стохастические переменные (это, кстати, заметно ускоряет процесс сэмпирования).

```
## One Harmonic (Generation + Reconstruction)
N = 256
Omega_truth = 0.2
A_truth, Phi_truth, Noise_truth = 2., 1., 5.
t = np.arange(0, N)
data0 = A_truth * np.cos(Omega_truth * t + Phi_truth)
data = data0 + Noise_truth * norm.rvs(size=N)
plt.plot(t, data)

with pm.Model() as OneHarmonicModel:
    Amp = pm.Uniform("Amp", lower=0.5, upper=5)
    Omega = pm.Uniform("Omega", lower=0.1, upper=0.3)
    Phi = pm.VonMises("Phi", mu=0., kappa=0.01)
    log_sigma = pm.Uniform("log_sigma", lower=-0.1, upper=10.0)
    Sigma = pm.Deterministic("Sigma", pm.math.exp(log_sigma))
    #Sigma = pm.HalfNormal("Sigma", sigma=10)
```

```

pm.Normal("obs", mu=Amp*pm.math.cos(Omega*t+Phi), sigma=Sigma, observed=data)
trace = pm.sample(tune=2000, draws=4000, chains=4, target_accept=0.97 )

az.plot_trace(trace, var_names=["Omega", "Sigma", "Amp", "Phi"]);
az.summary(trace)

## Two Harmonics (Generation + Reconstruction)
N = 256
f1, f2 = 1./10, 1./14
A1, A2 = 6., 3.
Phi1_truth, Phi2_truth = 0.5, 1.0
Noise_truth = 5.
t = np.arange(N)
data2 = A1*np.cos(2*np.pi * f1*t + Phi1_truth) + A2*np.cos(2*np.pi*f2*t+Phi2_truth)
data2 = data2 + norm.rvs(random_state=42, size=N, scale=Noise_truth)
plt.plot(t, data2)

with pm.Model() as TwoHarmonicsModel:
    log_sigma = pm.Uniform("log_sigma", lower=-0.1, upper=10.0)
    Sigma = pm.Deterministic("Sigma", pm.math.exp(log_sigma))
    A1 = pm.HalfNormal("A1",6)
    A2 = pm.HalfNormal("A2",6)
    T1_0 = pm.Exponential("T1_0", 1./5)
    T2_0 = pm.Exponential("T2_0", 1./5)
    T1 = pm.Deterministic("T1", T1_0 + 2.)
    T2 = pm.Deterministic("T2", T2_0 + 2.)
    Phi1 = pm.VonMises("Phi1", mu=0., kappa=0.01)
    Phi2 = pm.VonMises("Phi2", mu=0., kappa=0.01)
    S = A1*pm.math.cos(2*np.pi/T1*t+Phi1) + A2*pm.math.cos(2*np.pi/T2*t+Phi2)
    l = pm.Normal("LKH", mu=S, sigma=Sigma, observed=data2)
    trace2 = pm.sample(draws=2000, tune=5000, target_accept=0.97)

az.plot_trace(trace2, var_names=["T1", "T2", "Sigma", "A1", "A2", "Phi1", "Phi2"]);
az.plot_pair(trace2, var_names=["T1", "T2"])

```

6 Определение числа периодических компонент

TODO

7 Анфолдинг

Широкий класс исследовательских задач связан с процедурой *анфолдинга* (Unfolding) – получением по гистограмме измеренных данных гистограммы в пространстве интересующих ученого переменных. Фактически, речь идет о варианте обратной задачи по бинаризованным данным. Один из наиболее распространенных подходов к анфолдингу был сформулирован в 1995 году G.D. Agostini в статье A multidimensional unfolding method based on Bayes' theorem (из названия понятен и наш интерес) и позже программно реализован в виде пакета `PyUnfold`. На сайте проекта есть довольно простые примеры анфолдинга (к сожалению, только на элементарных модельных данных): <https://jrbourbeau.github.io/pyunfold/examples.html>. Кроме оригинальной статьи, математическое обоснование данного подхода подробно сформулировано вот в этой заметке James Bourbeau, Zigfried Hampel-Arias: https://github.com/jrbourbeau/pyunfold/blob/master/docs/latex_writeup/main.pdf.

На самом деле тема анфолдинга гораздо шире и ей даже был посвящен один из последних воркшопов PHYSTAT – [France-Berkeley PHYSTAT conference on Unfolding](#) (июнь 2024 г., Париж). Кроме того, в статье Georgios Choudalakis, [Fully Bayesian Unfolding](#) приведена байесовская формулировка задачи анфолдинга (несмотря на свое название методика G.D. Agostini таковой не является).

Анфолдинг пришел на смену широко распространенному методу bin-to-bin correction, основанного на Монте-Карло оценке обобщенной эффективности (она может быть даже больше единицы), вычисленной как отношение количества событий, попадающих в определенный бин реконструированной переменной, и количества событий в том же бине истинной переменной – эта эффективность затем используется для корректировки количества событий, наблюдаемых в бине. Очевидно, что данный метод требует одинакового разбиения на бины истинной и экспериментальной переменной и, следовательно, не может учитывать большие миграции событий из одного бина в другие. Более того, он пренебрегает неизбежными корреляциями между данными в соседних бинах.

Предположим, мы хотим исследовать некоторое явление, описываемое количественной характеристикой C (в общем случае многомерной). Для этого мы ставим эксперимент и проводим измерение другой величины E , связанной с C каким-то простым образом. Например, линейным. На языке `PyUnfold` C называется *cause* («причиной»), а E – *effect* («эффектом»). Точнее, мы будем описывать тот весьма распространенный случай, когда измерения бинаризованы, т.е. каждое измеренное событие попадает в один из бинов E_j , $j = 1, \dots, N_E$. Предположим, связь между C и E известна (например, в результате отдельного калибровочного эксперимента), и в результате нашего основного эксперимента мы получили распределение событий по бинам эффектов $\{n(E_j)\}$. Можем ли мы пересчитать его в причины C , а именно, считая величину причины тоже бинаризованной, получить распределение числа событий с тем или иным значением причины $\{\phi(C_\mu)\}$, $\mu = 1, \dots, N_C$, здесь N_C – количество бинов, на которые мы разбили ось C .

Казалось бы, что может быть проще. Если известно линейное преобразование от C к E , которое в бинаризованных данных можно записать в виде

$$E_j = \sum_{\mu=1}^{N_C} R_{j\mu} C_\mu$$

(в общем случае можно еще рассмотреть и сдвиг, связанный, например, с фоновыми отсчетами), то достаточно просто обратить *матрицу отклика* (response matrix) R . Кстати, в таком формате можно представить даже задачу *ре-биннинга*, когда $C \equiv E$, но требуется

построить гистограмму с более мелкими бинами. В задачах формирования изображения такой ребининг получил названия *суперразрешения*. Конечно, для получения суперразрешения одних гистограммных данных будет недостаточно – нужна еще дополнительная (априорная!) информация.

Но во многих интересных задачах матрица отклика может оказаться не квадратной, сингулярной или плохо обусловленной. В этих случаях зашумленные данные практически бесполезно обращать таким непосредственным образом. Потребуется *вероятностный вывод* (probabalistic inference), основанный на теореме Байеса. Перейдем к формулировке такого подхода.

Пусть было зарегистрировано некоторое событие – для определенности обозначим его номером k – и оно попало в бин E_j по измеренному эффекту. Поставим вопрос: какова вероятность того, что это событие имело причину из бина C_μ ? Ответ дает теорема Байеса:

$$p(C_\mu|E_j) = \frac{p(C_\mu)p(E_j|C_\mu)}{\sum_{\nu=1}^{N_C} p(C_\nu)p(E_j|C_\nu)},$$

где мы расписали знаменатель явным образом.

Переведем этот результат на язык гистограмм измерений $\mathcal{D} = \{n_j\}$, где $n_j \equiv n(E_j)$ – количество зарегистрированных событий с эффектом в бине E_j . Другими словами, мы K раз «подбрасывали кубик» с N_E гранями и он n_j раз выпал гранью j кверху. Тогда в величине

$$\hat{\phi}(C_\mu) = \sum_{j=1}^{N_E} p(C_\mu|E_j) n_j$$

несложно увидеть постериорную оценку среднего биномиального распределения...*SSh: На этом я приостанавливаю анализ алгоритма PyUnfold, так как он не до конца понятен, а его формулировка неоднозначна. Скорее всего, его итерационная процедура эквивалентна применению EM-алгоритма к задачам MAP-оценки. Но надо сначала самому в этом разобраться.*

Подойдем к этой задаче со стороны вероятностного программирования и ее реализации средствами PyMC. В этом случае нам придется ввести в модель следующие стохастические переменные: $\phi = \{\phi_\mu\}$ – число событий в гистограмме причин, $n = \{n_{j\mu}\}$ – число событий в каждом бине эффекта, вызванных каждой причиной. Но данные измерений n_{obs} у нас сформулированы не в виде n , а в виде их суммарных значений $\sum_\mu n_{j\mu}$. Оказывается для такой формулировки можно воспользоваться операцией sum (сформировав детерменистическую переменную) и функцией observe:*SSh: Здесь и далее «сидит» какая-то ошибка - я пока не разобрался с предлагаемой мной же процедурой. Более последовательный байесовский подход изложен далее в разделе, посвященном FBU.*

with pm.Model() as m:

```
phi = pm.Multinomial("phi", n = N_C, p = theta_C)
n = pm.Multinomial("n", n = phi, p = theta_EC)
n_histo = pm.Deterministic("n_histo", pm.math.sum(n, axis=2))
```

```
new_m = pm.observe(m, {n_histo: obs_data})
pm.sample(model = new_m)
```

В эту модель должны быть переданы:

- N_C – количество бинов, на которые разбита ось причин C ($\mu = 1, \dots, N_C$);

- $\theta_C \equiv \{\theta_\mu = p(C_\mu)\}$ – априорная вероятность того, что событие принадлежит бину C_μ ($\sum_\mu \theta_\mu = 1$);
- $\theta_{EC} \equiv \{\theta_{j\mu} = p(E_j|C_\mu)\}$ – калибровочная вероятность того, что событие из бина C_μ в результате измерения попадает в бин эффекта E_j ($\sum_j \theta_{j\mu} = 1$);
- $\text{obs_data} = \{n_j^{\text{obs}}\}$ – измеренное распределение событий по шкале эффектов E ($j = 1, \dots, N_E$).

При таком подходе подразумевается, что наблюдаемые значения n_j^{obs} получаются точными, без «ошибок измерения». Это не совсем то, что бы мы хотели на практике. Поэтому желательно ввести в модель флуктуации, например – пуассоновские:

```
with pm.Model() as m_P:
    phi = pm.Multinomial("phi", n = N_C, p = theta_C)
    n = pm.Multinomial("n", n = phi, p = theta_EC)
    n_histo = pm.Deterministic("n_histo", pm.math.sum(n, axis=2))
    pm.Poisson("n_obs", mu = n_histo, observed = obs_data)
```

```
pm.sample(model = m_P)
```

Обобщение на другие модели ошибок измерения очевидно. *SSh: Надо разобраться здесь с размерностями стохастических тензоров*

7.1 Fully Bayesian Unfolding

Разберем подробно одноименную [статью](#) G.Choudalakis, переформатировав ее под приведенные выше обозначения PyUnfold (и убрав разделы, посвященные тонкостям сэмплинга).

Начнем, пожалуй, с небольшого словарика обозначений:

- C – количественная характеристика, «причина» (cause, в общем случае многомерная) изучаемого явления. Нас интересует распределение событий по C – «спектр», в дальнейшем он будет фигурировать как дифференциальное распределение $d\phi/dC$.
- C_μ – событие, для которого характеристика C попадает в бин с номером μ . То есть изначально мы по той или иной причине интервал значений интересующей нас переменной разбили на N_C бинов, $\mu = 1, \dots, N_C$, и в конечном итоге будем интересоваться не непрерывным распределением по C , а огрубленным по дискретным бинам (их иногда тоже будем обозначать C_μ).
- E – величина, получаемая в результате измерений. Точнее, опять же по каким-то причинам само значение E для исследуемого нами события нам не дано, а известно только в какой бин оно попадает.
- E_j – событие, в результате измерений попавшее в бин j . Чтобы сформулировать полную систему (взаимоисключающих) событий, добавим сюда еще событие E_0 , означающее, пропуск события (т.е. в результате измерения событие не было зарегистрировано!)

- $\phi_\mu \equiv \phi(C_\mu)$ – истинное значение спектра, понимаемое как ожидаемое число событий в бине μ (“true-level” spectrum, the number of events expected to be produced in C_μ). В принципе, если знать распределение по непрерывной переменной C , $d\phi/dC$, то легко пересчитать его в бинаризованный спектр ϕ_μ (как интеграл по бину).
- $n_j \equiv n(E_j)$ – измеренный спектр, т.е. число событий, попавших в результате измерений в бин $j = 1, \dots, N_E$ (the number of events observed in bin E_j). Обратите внимание: у нас нет данных по $n_0 \equiv n(E_0)$.
- $\mathcal{M}_{j\mu} \equiv p(E_j, C_\mu)$ – матрица миграций (migration matrix, joint probability of an event to be produced in C_μ and reconstructed in E_j).
- $\mathcal{R}_{j\mu} \equiv p(E_j|C_\mu)$ – матрица отклика (response matrix, conditional probability for an event that was produced in C_μ to be reconstructed in E_j).

По своему определению $p(E_j|C_\mu) = p(E_j, C_\mu)/p(C_\mu)$ и $p(C_\mu) = p(E_0, C_\mu) + \sum_{j=1}^{N_E} p(E_j, C_\mu)$. Поэтому, обозначая $p(E_0|C_\mu) \equiv 1 - \epsilon_\mu$, для *эффективности регистрации* ϵ_μ имеем выражение

$$\epsilon_\mu = \frac{\sum_{j=1}^{N_E} p(E_j, C_\mu)}{p(C_\mu)} = \frac{\sum_{j=1}^{N_E} \mathcal{M}_{j\mu}}{p(C_\mu)}$$

Предполагается, что матрица \mathcal{M} была заранее получена в результате калибровки измерительного прибора.

Байесовская постановка задачи анфолдинга заключается в получении постериорного распределения $p(\Phi|\mathcal{D})$ по априорному распределению $p(\Phi)$, где для упрощения записи введены обозначения:

$$\begin{aligned} \Phi &\equiv \{\phi_\mu\}, \quad \mu = 1, \dots, N_C, \\ \mathcal{D} &\equiv \{n_j\}, \quad j = 1, \dots, N_E. \end{aligned}$$

На основании теоремы Байеса, $p(\Phi|\mathcal{D}, \mathcal{M}) \propto p(\mathcal{D}|\Phi, \mathcal{M}) \cdot p(\Phi, \mathcal{M})$.

По какой-то причине автор FBU здесь и далее обуславливается по матрице миграции \mathcal{M} , а не по матрице отклика \mathcal{R} . И все дальнейшее изложение статьи на это опирается. Автор даже вводит понятие True MC spectrum и Actual MC spectrum (и четко различает их). Лично мне представляется, что лишь матрица отклика \mathcal{R} может быть получена в рамках калибровочного эксперимента. Тогда какой смысл имеет обуславливание по \mathcal{M} ?!

Какой может быть функция правдоподобия $p(\mathcal{D}|\Phi, \mathcal{M})$?

Самый простой вариант: при заданных Φ и \mathcal{M} считать все n_j независимыми одинаково распределенными (i.i.d.) величинами. Более того, в силу их определения как числа событий имеет смысл считать их пуассоновскими, т.е.

$$p(\mathcal{D}|\Phi, \mathcal{M}) = \prod_{j=1}^{N_E} p(n_j|\Phi, \mathcal{M}) = \prod_{j=1}^{N_E} \frac{r_j^{n_j}}{n_j!} \exp\{-r_j\},$$

где r_j имеет смысл ожидаемого в результате реконструкции числа событий в бине (the number of events expected to be reconstructed in bin E_j). Очевидно, что

$$r_j = \sum_{\mu=1}^{N_C} \phi_\mu p(E_j|C_\mu) = \sum_{\mu=1}^{N_C} \phi_\mu \mathcal{R}_{j\mu} \quad (10)$$

Ох, как меня эти «очевидно» пугают: есть ощущение, что это утверждение ниоткуда не следует, а является фактически формулировкой предлагаемой модели. Надо будет глубже разобраться с этим (и с предлагаемой пуассоновостью величины n_j). Пока так: r_j – набор скрытых (латентных) переменных модели, определяемых в соответствии с (10) и фигурирующих в функции правдоподобия... Кстати, из определения непосредственно видно, что для распределения \mathcal{D} достаточно обусловиться по Φ и \mathcal{R} (а не по \mathcal{M})!

А что на счет прайора $p(C_\mu)$?

Если считать, что наряду с матрицей миграции \mathcal{M} в результате калибровки заданы и эффективности ϵ_μ , то $p(C_\mu) = \frac{1}{\epsilon_\mu} \sum_{j=1}^{N_E} \mathcal{M}_{j\mu}$. А можно задать само распределение $p(C_\mu)$ (например, исходя из каких-то дополнительных – априорных – знаний о структуре распределения) и уже на основе этого вычислять эффективности. В любом случае $\mathcal{R}_{j\mu} = \epsilon_\mu \mathcal{M}_{j\mu} / (\sum_{i=1}^{N_E} \mathcal{M}_{i\mu})$, и поэтому

$$r_j = \sum_{\mu=1}^{N_C} \frac{\phi_\mu \epsilon_\mu \mathcal{M}_{j\mu}}{\sum_{i=1}^{N_E} \mathcal{M}_{i\mu}} + b_j, \quad (11)$$

где добавлением последнего слагаемого мы обобщили ситуацию на наличие фоновых отсчетов с ожидаемым количеством b_j .

Сейчас у меня такое представление. Все-таки на основании калибровки прибора мы получаем именно матрицу отклика \mathcal{R} (и эффективности следуют из нее как $\epsilon_\mu = \sum_{j=1}^{N_E} \mathcal{R}_{j\mu}$), а для проведения ВІ, как обычно, требуется еще выбрать тот или иной прайор $p(C_\mu)$. С точки зрения автора FBU эти прайоры являются «вероятностными» регуляризаторами, и он подробно обсуждает их влияние на результат реконструкции. А ситуация, когда прайор совпадает с «истинным», необходима ему для объяснения эксперимента по открытию бозона Хиггса: есть теоретическое предсказание, которое обосновывает выбор такого прайора, и нам требуется реконструировать экспериментальные данные, опираясь на эту теорию... Но что-то меня тут все-таки смущает.

В самой статье рассматривается ситуация с модельным истинным спектром вида

$$\frac{d\phi}{dC} = (1 - C/C_{\max})^6 (C/C_{\max})^{-4.8}, \quad C_{\max} = 7000, \quad (12)$$

навязанным массовым dijet-спектром в протонных столкновениях при энергии $\sqrt{s} = 7000$ GeV (все C в GeV). Калориметрические измерения хорошо описываются как

$$E = C + \sigma(C) \cdot \xi, \quad \xi \sim \mathcal{N}(0, 1) \quad (13)$$

с упрощенной версией энергетического разрешения

$$\sigma(C) = C \left(\frac{a}{\sqrt{C}} + b \right) \quad (14)$$

Параметры a , b получаются в результате калибровки, в самой статье для анализа использовались значения $a = 0.5$, $b = 0.1$. Отсутствию размытия данных соответствуют $a = b = 0$.

Как была получена матрица миграции?

Для этого генерировалось $K = 10^7$ событий C_k в соответствии с (12), после чего на основании этого массива событий составлялась гистограмма $\phi(C_\mu)$ с разбиениями по $N_\mu = 14$ бинам $C_\mu = 500 \exp\{0.15\mu\}$ (по какой-то причине⁶ при составлении гистограммы каждому событию дополнительно назначался вес $w_k = 10^{-3}$). Кроме того, для каждого C_k на основании (13) рассчитывалось E_k и составлялась гистограмма $n(E_j)$ с N_E бинами E_j (в данной работе они выбирались идентичными бинам C_μ – таким образом, «исчезнувшим» событиям E_0 соответствовали лишь некоторые из «краевых» событий C_k). На основании этого значения $M_{j\mu}$ определялись как $n(E_j)\phi(C_\mu)/K^2$. *SSH: Последнее утверждение не верно - см. пояснения далее!*

Вопрос: так полученная матрица миграции зависит непосредственно от исследуемого спектра ϕ – разве это можно назвать калибровкой?

По самому смыслу калибровка подразумевает оценку отклика прибора на измеряемый сигнал. Так что в результате калибровки на самом деле мы можем получить матрицу отклика \mathcal{R} (и коэффициенты эффективности как $\epsilon_\mu = \sum_{j=1}^{N_E} \mathcal{R}_{j\mu}$). Поэтому, вроде как, правильнее было бы генерить события C_k из произвольного распределения $p(C)$, и на основании этого составлять гистограммы $\phi(C_\mu)$ и $n(E_j)$ и уже по ним вычислять матрицы \mathcal{M} и \mathcal{R} (снова $M_{j\mu} = n(E_j)\phi(C_\mu)/K^2$, а вот $\mathcal{R}_{j\mu} = M_{j\mu}/p(C_\mu)$, где в знаменателе стоит $p(C_\mu) = \int_{C_\mu} p(C)dC$). Почему автор данной статьи (да и многие другие) так не делают – большой вопрос! *SSH: СДЕЛАТЬ САМОМУ!* Ответ, наверное, заключается в том, что матрица отклика \mathcal{R} , по идее, не должна зависеть от выбранного распределения $p(C)$ – почему б тогда не воспользоваться тем, что исследуется в задаче...

SSH: ПРОДОЛЖЕНИЕ СЛЕДУЕТ

SSH: Так, кажется я нашел ошибку в своих же рассуждениях. Попробую пояснить...

В приведенном выше генерационном алгоритме расчета матриц \mathcal{M} и \mathcal{R} закралась весьма существенная ошибка. Утверждение $M_{j\mu} = n(E_j)\phi(C_\mu)/K^2$ не верное, так как фактически подразумевает факторизацию $p(E_j, C_\mu) = p(E_j)p(C_\mu)$ (и «удивительное» редуцирование $\mathcal{R}_{j\mu}$ до n_j/K). Правильным же является поиск элементов $M_{j\mu}$ как отсчетов двумерной гистограммы, построенной по данным (C_k, E_k) в бинах C_μ, E_j . Тогда факторизации и редукции не происходит!

Уточним эту, новую, процедуру. Пусть $m_{j,\mu}$ – количество событий (E_k, C_k) , $k = 1, \dots, K$, попавших в бин j, μ двумерной гистограммы. По определению $\sum_\mu m_{j,\mu} = n_j$, $\sum_j m_{j,\mu} = \phi_\mu$ и

$$\mathcal{M}_{j\mu} = \frac{m_{j,\mu}}{K}, \quad \mathcal{R}_{j\mu} = \frac{\mathcal{M}_{j\mu}}{p(C_\mu)} = \frac{m_{j,\mu}}{\sum_i m_{i,\mu} + m_{0,\mu}}$$

Обращаем внимание, что в знаменателе последней формулы суммирование ведется как по «сигнальным» индексам $i = 1, \dots, N_E$, так и по пропущенным событиям (не-эффективность канала регистрации⁷). В пределе $K \rightarrow \infty$ так полученные значения $\mathcal{R}_{j\mu}$ не должны зависеть от выбора «первичного» спектра $p(C)$. Интересно, а автор FBU точно так делал?!

Основные задачи, рассмотренные в статье FBU:

- 6.1 No smearing, 2 bins, high statistics: $a = b = 0$, $N_C = N_E = 2$, Figs 3, 4, 5. $\Rightarrow p(\phi_1, \phi_2|\mathcal{D})$ – симметричное гауссо-подобное распределение.

⁶Пока единственное объяснение этому факту (в статье оно не приводится) – чтобы сделать сопоставимыми масштабы распределения $d\phi/dC$ и гистограммы $\phi(C_\mu)$.

⁷По самому определению алгоритма у нас стопроцентная эффективность канала причины – мы называем событие *событием* только, если оно попало в какой-то из событийных бинов C_μ , $\mu = 1, \dots, N_C$. А вот канал регистрации может быть не эффективным. Правда, в рассматриваемой в статье модели измерения эта неэффективность тривиальна и связана лишь с попаданием на края спектра.

- 6.2 No smearing, 2 bins, low statistics: $a = b = 0$, $N_C = N_E = 2$, $K \rightarrow K/10^3$, Figs 6, 7, 8.
 $\Rightarrow p(\phi_1, \phi_2 | \mathcal{D})$ – не симметричное пуассоно-подобное распределение.
- 6.3 More smearing, in 2 bins: $a = 0$, $b = 0.1, 0.3, 0.5, 0.8$, $N_C = N_E = 2$, Figs 9, 10, 11.
- 6.4 Smearing at $N_C = N_E = 14$: $a = 0.5$, $b = 0.1$, Figs 1(b), 2, 12, 13, 14, 15, 16, 18.
- 6.5 Smearing with $N_E \neq N_C$: $a = 0.5$, $b = 0.1$, $N_C = 14$, $N_E = 5$, Figs 19, 20, 21, 22.
- 6.6 Unfolding a bump: Model with constant spectrum and fixed smearing $\sigma = 0, 50, 75, 100, 150$ and $N_C = 30$, an expected bump – Figs 23, 24, an unknown bump – Figs 25, 26.

Как можно обобщить модель (13)?

Масштабирование переменной C тривиально. Поэтому рассмотрим более интересный сценарий, позволяющий учесть «настоящую» не-эффективность канала регистрации, а не только ее краевую составляющую. В принципе, все просто. При генерации событий C_k и их пересчете в E_k с определенной вероятностью $1 - \epsilon$ засчитываем это событие как *пропущенное* (т.е. приписываем ему номер 0). Эту самую эффективность можно задать как функцию исходной переменной C , $\epsilon(C)$, а можно привязать непосредственно к бину C_μ , $\epsilon = \epsilon(C_\mu)$. Дальнейшая работа генерационного алгоритма остается без изменений.

Примечание. В принципе на задачу байесовского анфолдинга можно смотреть и как на пересчет одного спектра к другому. Ведь при калибровке мы также имеем дело со спектром – спектром *калибровочного сигнала* $\phi^{(c)}$, который, конечно же, не то же самое, что исследуемый нами спектр (неизвестного) источника ϕ . По своему определению калибровочный спектр $\phi^{(c)}$ мы знаем с высокой точностью, иначе странно было бы эти предварительные измерения называть калибровочными. Калибровочные измерения нужны нам, чтобы получить матрицу отклика \mathcal{R} нашей измерительной аппаратуры. В рамках байесовского подхода следует эту матрицу считать стохастической и тогда калибровочный эксперимент – просто способ уточнить свои априорные знания о \mathcal{R} .

Таким образом, у нас есть два спектра и два набора наблюдаемых данных, $\mathcal{D} = \{n_j\}$ и $\mathcal{D}^{(c)} = \{n_j^{(c)}\}$, и модель, структурирующая наши представления о характере измерений. Априорно об одном из спектров, сигнальном ϕ , мы мало что знаем – поэтому и проводим эксперимент по набору данных \mathcal{D} . А вот калибровочный спектр $\phi^{(c)}$ должен быть нам известен очень хорошо – ведь именно за счет этого мы получим информацию об аппаратной функции детектора \mathcal{R} . В результате ВІ мы хотим уточнить информацию как об \mathcal{R} , так и о спектре сигнала ϕ .

Вопрос: есть ли оптимальное разбиение шкалы причин на бины C_μ ? Разбиение на бины шкалы измерений определяется самой измерительной аппаратурой – и с этим мы уже ничего поделать не можем. А вот разбиение на бины шкалы C в некотором смысле произвольно. Кто нам мешает сделать эти бины мельче или выбрать их границы каким-нибудь нестандартным (например, не эквидистантным) способом. Понятно, что при очень мелком разбиении наблюдаемых данных может не хватить, чтобы дать надежную оценку ϕ_μ – этим обычно и вызвано бинаризация сигнала. И все же: есть ли какой-то оптимум в выборе разбиения C_μ и от чего он в принципе зависит? Причем к этому непростому вопросу можно подойти и в двух разных сценариях: 1) с заранее хорошо известной матрицей отклика $\mathcal{R}_{j\mu}$ и 2) когда разбиение на C -бины может повлиять и на сам процесс калибровки \mathcal{R} .

В данной работе мы будем различать два вида спектров:

- 1) Физический спектр $d\phi_{\text{phys}}(C)/dC$ и его нормированную версию $p_{\text{phys}}(C)$.
- 2) Калибровочный спектр $d\phi_{\text{cal}}(C)/dC$ и его нормированную версию $p_{\text{cal}}(C)$.

Эти спектры иногда будет удобно представлять в бинаризованном виде: $p_{\text{phys}}(C_\mu) = \int_{C_\mu} p_{\text{phys}}(C) dC$ и $p_{\text{cal}}(C_\mu) = \int_{C_\mu} p_{\text{cal}}(C) dC$, где $\{C_\mu\}$ – разбиение на бины шкалы C .

Кроме этого, в задаче восстановления спектра будут фигурировать априорное и постериорное распределения бинаризованной версии физического спектра $p(\Phi)$ и $p(\Phi|\mathcal{D})$, где $\Phi \equiv \{\phi_\mu\}$, $\mu = 1, \dots, N_C$.

Обращаем внимание, что в имитационном эксперименте физический спектр (дифференциальный или бинаризованный) мы используем при генерации данных, а калибровочный – для оценки матрицы отклика детектора \mathcal{R} . И обычно эти спектры сильно отличаются друг от друга: один нам заранее не известен, а второй мы обязаны знать хорошо, чтобы откалибровать прибор!

А вот структура априорного распределения $p(\Phi)$ находится полностью под нашим контролем. Фактически в этом и заключается вероятностное моделирование эксперимента: мы пытаемся интерпретировать полученные данные за счет структур, вложенных в модели $p(\Phi)$ и $p(\mathcal{D}|\Phi)$. Конечно, задачу реконструкции всегда можно сформулировать так, что функциональный вид спектра нам известен, и задача интерпретации заключается лишь в уточнении фигурирующих в модели спектра параметров. Но это совсем не обязательно: можно остаться в ситуации free-form solution и пытаться получить $p(\Phi|\mathcal{D})$ в наиболее общем непараметризованном виде.

Генерация данных

- 1) Задавшись спектром $d\phi_{\text{phys}}/dC$ и интервалом $[C_{\min}, C_{\max}]$, генерим K событий C_k , $k = 1, \dots, K$.
- 2) Для каждого C_k , воспользовавшись *моделью размытия* (smearing model), вычисляем значение эффекта $E_k = R[C_k]$. Модель размытия может быть как детерминированной (например, просто свертка – конволюция – с аппаратной функцией) или стохастической (так, в статье FBU используется $R[C] \equiv C + \sigma(C)\xi$, $\xi \sim \mathcal{N}(0, 1)$), в общем случае параметризованной (в рамках генерационной модели параметры размытия фиксированы).
- 3) Задавшись разбиением шкал причины $\{C_\mu\}$, $\mu = 1, \dots, N_C$, и эффекта $\{E_j\}$, $j = 1, \dots, N_E$, рассчитываем гистограммы частот попадания в соответствующие бины величин C_k и E_k :

$$\Phi \equiv \{\phi_\mu\} = \text{histo}[\{C_k\}, \{C_\mu\}]$$

и

$$R \equiv \{r_j\} = \text{histo}[\{E_k\}, \{E_j\}]$$

- 4) Генерим данные в соответствии с выбранной моделью ошибок измерения. Для детекторов типа счетчиков целесообразно выбрать независимые пуассоновские флуктуации сигнала:

$$n_j \sim \text{Poisson}(r_j + b_j)$$

где $B \equiv \{b_j\}$ – дополнительный фон в измерительном канале (также пуассоновский и независимый от основного сигнала).

Калибровка

- 1) Задавшись спектром $d\phi_{\text{cal}}/dC$ и интервалом $[C_{\min}, C_{\max}]$, генерим K_c событий C_k^c , $k = 1, \dots, K_c$.
- 2) В соответствии с выбранной моделью размытия рассчитываем значения эффекта $E_k^c = R[C_k^c]$. Модель размытия должна быть той же самой, что и на этапе генерации данных (ведь мы калибруем тот самый прибор!), а вот спектр событий (калибровочный) может (и должен) отличаться от физического. Главное, чтобы он нам был хорошо известен (о значении термина «хорошо» см. ниже). *SSH: Зачем это?!*

3) Оцениваем элементы матрицы отклика \mathcal{R} как

$$\mathcal{R}_{j\mu} = \frac{m_{j,\mu}}{\sum_i m_{i,\mu} + m_{0,\mu}}$$

где $\{m_{j,\mu}\}$ – двумерная гистограмма, получаемая бинаризацией данных (E_k^c, C_k^c) по бинам $\{E_j, C_\mu\}$ (т.е. тем же самым, что фигурировали при генерации данных):

$$M \equiv \{m_{j,\mu}, m_{0,\mu}\} = \text{histo2d}[\{(E_k^c, C_k^c)\}, \{E_j, C_\mu\}].$$

Здесь $m_{0,\mu}$ обозначено число пропусков – событий C_k^c из бина C_μ , не попавших ни в один из бинов E_j , $j = 1, \dots, N_E$.

4) А как вычислить неопределенность матрицы отклика \mathcal{R} ? Желательно убедиться, что при достаточно больших K_c так полученные значения $\mathcal{R}_{j\mu}$ не зависят от выбора калибровочного спектра. Однако на практике мы никогда не знаем матрицу отклика абсолютно точно – всегда есть та или иная степень неопределенности в ее коэффициентах. Это может быть связано с точностью измерений в раках калибровочного эксперимента, а может иметь и модельную природу (ведь, по сути, параметризация отклика детектора матрицей это всего лишь приближение). Поэтому крайне важно придумать способ учета неопределенности в матрице отклика. Пока же будем считать ее заданной точно...

Реконструкция

1)

У Georgios Choudalakis (автора FBU) в 2011 году вышел препринт [How to Use Experimental Data to Compute the Probability of Your Theory](#). В этой работе он детально (с образовательным уклоном) разбирает применение байесового вывода в задачах счетчика пуассоновских данных. Рекомендую для разбора на семинаре БАД.

8 Задача Павла Волчугова

При детектировании ШАЛ с помощью Черенковский атмосферных телескопов, изображение на фокальной поверхности оптической системы представляет собой овалоподобную структуру, которую мы для простоты в дальнейшем будем называть просто «эллипс». При регистрации события несколькими телескопами с совпадающими полями зрения все эллипсы можно привести к единой системе координат $X - Y$. Каждый из них может быть охарактеризован положением центра (x_i, y_i) и его ориентацией, выраженной азимутальным углом ϕ_i . Геометрически направляющие оси этих эллипсов (прямые, проходящие через их центр) должны пересекаться в одной точке (x_0, y_0) – точке пересечения оси ливня с фокальной плоскостью (ФП), единой для всех телескопов. Ввиду наличия погрешностей в поредлении параметров эллипсов (а также возможных ошибках в калибровке полей зрения), оси измеренных эллипсов не пересекаются в одной точке (при $N > 2$), и возникает задача определения наиболее правдоподобной точки пересечения. Вероятностный подход на основании теоремы Байеса – наиболее прямой способ решения такой задачи.

Для того, чтобы приступить к байесовскому моделированию, сформулируем основные «ингредиенты» такого подхода – имеющиеся данные \mathcal{D} и параметры модели Θ .

В качестве данных выступают измеренные значения координат центров эллипсов и их ориентация

$$\mathcal{D} = \{x_i, y_i, \phi_i\}, \quad i = 1, \dots, N,$$

где N – количество изображений-эллипсов.

Примечание. Заметим, что сами эти значения получаются в результате анализа исходных изображений и в этом смысле являются не прямыми, а производными (предобработанными) данными. Отсюда сразу возникает понимание направления последующего усложнения модели: использование в качестве данных самих изображений, а не их параметризаций. Еще одним способом добавления информации в данные является учет также эксцентриситета эллипса (параметра его «сплюснутости») и его характерной амплитуды (той или иной меры интенсивности, например, суммарной).

Так как основной задачей является определение точки пересечения $A(x_0, y_0)$ «идеальных» осей эллипсов, то в число параметров модели, помимо этих координат, целесообразно включить и дополнительные параметры, связывающие основные с результатами измерений. Очевидно, что для этого достаточно параметризовать модель с помощью расстояний ρ_i от A до центра C_i эллипса и соответствующего азимутального угла:

$$\Theta = \{x_0, y_0, \rho_i, \psi_i\}$$

Для идеальных изображений должны иметь

$$\begin{aligned} \tilde{x}_i(\Theta) &= x_0 + \rho_i \cos \psi_i \\ \tilde{y}_i(\Theta) &= y_0 + \rho_i \sin \psi_i \\ \tilde{\phi}_i(\Theta) &= \psi_i + \pi \end{aligned}$$

где было учтено, что направление оси эллипса противоположно направлению на его центр.

Для завершения построения байесовской модели необходимо ввести функцию правдоподобия, отражающие характерный процесс измерения и его ошибки. Вполне логичным будет считать измерения различных изображений независимыми, т.е.

$$p(\mathcal{D}|\Theta) = \prod_{i=1}^N p(x_i, y_i, \phi_i|\Theta)$$

Характер распределения ошибок измерения отдельного эллипса, конечно же, зависит от самой процедуры оценки его параметров. В простейшем случае предположим также их независимыми:

$$p(x_i, y_i, \phi_i | \Theta) = p(x_i | \tilde{x}_i(\Theta)) p(y_i | \tilde{y}_i(\Theta)) p(\phi_i | \tilde{\phi}(\Theta))$$

и распределенными одинаково – с характерной погрешностью σ измерения координат x, y и погрешностью σ_ϕ измерения углов ориентации ϕ . Значения σ и σ_ϕ должны либо представляться самой процедурой оценки параметров эллипса, либо получаться на основании байесовского анализа данных (см. ниже).

Программная реализация предложенной выше модели очевидна. Однако сделаем несколько важных замечаний. Во-первых, если пойти по пути назначения широких (неинформативных) прайоров на все параметры модели, то стоит иметь ввиду, что параметры ρ_i по своему смыслу положительны, а параметры ϕ 2π -периодичны. Для радиальных расстояний достаточно будет выбрать полу-нормальный прайор, а вот учет периодичности лучше всего осуществить с помощью распределения фон Мизеса ([pm.VonMises](#)). Это распределение является аналогом гауссового распределения на окружности $[-\pi, \pi]$ (с отождествленными границами) и также имеют два параметра μ и κ . Параметр μ по-прежнему характеризует концентрацию распределения (его моду и среднее), а параметр κ – его сосредоточенность. Малые значения κ приводят к широким 2π -периодичным распределениям, большие – к узким, сильно концентрированным вблизи моды. Мы используем значение $\kappa = 0.01$ для моделирования (квази)равномерного распределения.

Распределением фон Мизеса рекомендуется воспользоваться не только для задания прайоров на углы, но и для функции правдоподобия. Связь между κ и стандартным отклонением (погрешностью измерений) выражается посредством соотношения

$$\sigma_\phi^2 = 1 - I_1(\kappa)/I_0(\kappa) \approx 1/(2\kappa)$$

где $I_{0,1}(\kappa)$ – модифицированные функции Бесселя первого рода (функции Инфельда), и в последнем (приближенном) равенстве мы воспользовались их асимптотиками при $\kappa \rightarrow \infty$. Таким образом, значение $\kappa = 1/50$ соответствует $\sigma_\phi = 0.1 \approx 5.7^\circ$.

Важно отметить также, что распределения фон Мизеса определены на интервале $[-\pi, \pi]$, поэтому прежде чем передавать в правдоподобие измеренные значения ϕ (посредством аргумента `observed`) необходимо привести их к этому интервалу добавлением (вычитанием) подходящего значения, кратного 2π .

С учетом этих замечаний PyMC-реализация простейшей байесовой модели задачи о пересечении осей эллипсов выглядит следующим образом.

```
phi = (phi - np.array([180, 180, 180+360]) * np.pi/180.);
sigma = 0.5;
with pm.Model() as model1:
    # Priors
    x0 = pm.Normal("x0", mu=0, sigma = 1.0)
    y0 = pm.Normal("y0", mu=0, sigma = 1.0)
    rho1 = pm.HalfNormal("rho1", sigma=5.)
    rho2 = pm.HalfNormal("rho2", sigma=5.)
    rho3 = pm.HalfNormal("rho3", sigma=5.)
    psi1 = pm.VonMises("psi1", mu=0., kappa=0.01)
    psi2 = pm.VonMises("psi2", mu=0., kappa=0.01)
    psi3 = pm.VonMises("psi3", mu=0., kappa=0.01)
    kappa = 50;
```

```

#kappa = pm.HalfNormal("kappa", sigma = 50.)

# Likelihood
pm.Normal("x1id", mu=x0+rho1*pm.math.cos(psi1), sigma=sigma, observed=x1)
pm.Normal("y1id", mu=y0+rho1*pm.math.sin(psi1), sigma=sigma, observed=y1)
pm.Normal("x2id", mu=x0+rho2*pm.math.cos(psi2), sigma=sigma, observed=x2)
pm.Normal("y2id", mu=y0+rho2*pm.math.sin(psi2), sigma=sigma, observed=y2)
pm.Normal("x3id", mu=x0+rho3*pm.math.cos(psi3), sigma=sigma, observed=x3)
pm.Normal("y3id", mu=y0+rho3*pm.math.sin(psi3), sigma=sigma, observed=y3)
pm.VonMises("phi1id", mu = psi1, kappa = 10+kappa, observed= phi[0])
pm.VonMises("phi2id", mu = psi2, kappa = 10+kappa, observed= phi[1])
pm.VonMises("phi3id", mu = psi3, kappa = 10+kappa, observed= phi[2])

trace1 = pm.sample(tune=2000, draws=4000, chains=4, target_accept=0.95)

```

Здесь в закомментированном виде представлена и простейшая модификация модели – со стохастическим параметром κ . Кроме того, обратите внимание как параметризована передача этого аргумента в правдоподобие: сдвиг на 10 позволяет отодвинуть сэмплер от «опасной» области значений вблизи 0!

9 Задача Виктора Кудрявцева

Исследуется задача восстановления сигнала на входе SiPM по данным специальной измерительной системы. Аналоговая часть системы содержит в себе набор быстро переключающихся конденсаторов, совокупное действие которых может быть аппроксимировано фильтром специального «экспоненциального» вида.

Изначально было предложено восстанавливать входной (активный) сигнал с помощью «дифференциальной» формулы типа $A_k \propto d_k - bd_{k+1}$ (здесь d_k – измеренное на выходе схемы значение сигнала в момент времени k). Однако, при использовании такой процедуры к зашумленным данным была показана ее крайняя неэффективность.

Экспериментальная реализация данной схемы продемонстрировала еще одну проблему: выходной сигнал помимо белого шума содержит сильные флуктуации (выбросы), которые сложно объяснить гауссовым распределением. Возможно, существует и временная корреляция в шумовых отсчетах.

Все это свидетельствует о необходимости разработки альтернативной процедуры «инвертирования» измерительной схемы, учитывающей как особенности аппаратной функции, так и статистические свойства шума. Очевидно, что как раз байесовский подход может оказаться весьма перспективным для решения такого рода проблем.

Аналогичная задача рассматривалась в классическом учебнике Sivia по байесовским методам в разделах, посвященных решениям в свободной форме (глава 6) и в планировании эксперимента (глава 7). Эта же тема подробно разобрана в MDA – см. разделы 1.6 и 1.7.

В наиболее общей постановке речь идет о восстановлении сигнала произвольной формы $f(t)$ (free-form solution). При байесовском моделировании для задания таких функций используется разбиение на M бинов с последующей аппроксимацией кусочно-постоянной функцией: $f(t) \approx \sum_j a_j \delta(t - t_j)$, где $\delta(t)$ – функция-индикатор («прямоугольник») бина.

Сигнал с выхода измерительной схемы представляет собой выборку в виде эквидистантно расположенных во времени сэмплов $\{d_k\}$, $k = 1, \dots, N$. В простейшем случае измерение можно считать гауссовым, т.е. формулировать функцию правдоподобия как

$$p(\mathcal{D}|\Theta) = \prod_{k=1}^N \mathcal{N}(d_k | f_k, \sigma_k), \quad (15)$$

где «идеальное» значение сигнала на выходе f_k является сэмплом из аналоговой (интегральной) свертки входного сигнала $f(t)$ с аппаратной функцией измерительной системы $R(t)$:

$$f_k = \int f(t) R(t - t_k) dt$$

Заменяя $f(t)$ кусочно-постоянной аппроксимацией, получаем

$$f_k = \Delta t \sum_{j=1}^M a_j R(t_j - t_k),$$

где дополнительно предположена эквидистантность аппроксимации.

Обращаем внимание, что при таком подходе нет необходимости считать разбиения $\{t_k\}$ и $\{t_j\}$ заданными на одной и той же сетке. Первое из них задается аппаратной реализацией считывающей части измерительной схемы (АЦП), а второе находится в нашей власти и определяет степень детализации, которую мы хотим выявить во входном сигнале. В

байесовской постановке даже нет необходимости предполагать, что $M \leq N$: вероятностное решение существует даже когда число неизвестных превышает число уравнений (еще раз: байесовский вывод – не инверсия!).

Естественно, гауссова модель (15) весьма ограничена и, скорее всего, плохо описывает отмеченные выше флуктуации в выходном сигнале. Однако она может сыграть роль «нулевой» модели, модификации которой впоследствии приведет к более надежным результатам.

Основная идея, использованная Sivia для решения этой задачи, – подбор прайора на коэффициенты $\{a_j\}$ в соответствии с принципом максимума энтропии MaxEnt. Мотивировка этого – положительность коэффициентов (ввиду положительности $f(t)$) и осмысленность операций их суммирования (интегрирования $f(t)$). В каком-то смысле это малоинформативный прайор, отражающий только эти две черты решения в свободной форме. При этом прайор зависит от одного дополнительного параметра α

$$p(\{a_j\}) \propto \exp\{\alpha S\}, \quad S \equiv \sum_{j=1}^M (a_j - m_j - a_j \ln(a_j/m_j))$$

Здесь разбиение $\{m_j\}$ – мера Лебега, выступающая в роли предопределенного сигнала. В простейшем случае это равномерная мера, $m_j = 1/M$.

При последовательном байесовском подходе необходимо назначить (гипер)прайор на α и затем провести по нему маргинализацию. Однако, как отмечает Sivia, в ряде работ было показано, что при этом обнаруживается существенная проблема: результаты вывода зависят от степени пикселизации (выбора M). Сам Sivia в своем учебнике предлагает пойти полу-байесовским путем, заменяя задачу вывода на задачу оптимизации и подбирая специальным образом значение α (см. подробнее раздел 1.6.2 в MDA).

Нам же представляется перспективным детально проанализировать зависимость результатов от выбора M , сформулировав вероятностную модель в PyMC. Правда, для этого необходимо «научиться» задавать в PyMC такого рода MaxEnt прайоры.

А как же это сделать? К сожалению, как нетрудно убедиться, среди [встроенных в PyMC распределений](#) MaxEnt не существует (что весьма странно!). Поэтому надо как-то хитрить и попытаться реализовать подобный прайор. О том, как это делается в случае функции правдоподобия, подробно изложено в заметке [Using a “black box” likelihood function](#). Также полезной (но сложной!) может оказаться заметка [Approximate Bayesian Computation](#).

Одна из причин, по которой в PyMC не реализован MaxEnt, как нам кажется, заключается в том, что этот подход в некотором смысле, устарел. Его расцвет пришелся на 80-90-е годы прошлого века и был во многом вызван относительно слабыми вычислительными возможностями компьютеров тех лет. В частности, MaxEnt-процедура использовалась не в форме полноценного байесовского вывода, а в качестве оптимизационной задачи с ограничениями (энтропия как раз и выступала в качестве регуляризатора, подробнее см. раздел 1.6.2 документа MDA).

В 21-м веке на смену MaxEnt приходят другие, более сложно задаваемые прайоры и параметризации моделей. В частности, для решений в свободной форме в рамках полноценного Байеса могут применяться разложение по сплайнам – хорошую заметку об этом читайте здесь: [Splines](#).

Наконец, наиболее «продвинутым» подходом для решения задач в свободной форме (и привлечшим большое внимание специалистов) становятся так называемые *гауссовы процессы* (GP). На сайте разработчиков PyMC примерам по GP посвящен целый раздел

Gaussian processes. Разбор наиболее интересных, с нашей точки зрения, из них приведен ниже в Приложении A.4. Кроме того, я настоятельно рекомендую прочитать внимательно главу 7 книги Osvaldo Martin, *Bayesian Analysis with Python* (далее просто [Martin2024]).

9.1 Постановка задачи

Формализуем постановку задачи следующим образом. Пусть A_k , $k = 1, \dots, K$, обозначает скрытые переменные, равные ожидаемому значению сигнала на входе детектора. Для простоты ограничимся детекторами типа счетчиков, поэтому на A_k можно смотреть как на ожидаемое число фотонов (точнее, фотоэлектронов) за k -й интервал времени (такт работы электроники). Если $I(t)$ обозначить зависимость интенсивности входного сигнала от времени, то $A_k = \int_{t_k}^{t_k + \Delta t} I(t) dt$, где Δt – длительность такта.

Важно различать ожидаемое значение входного сигнала A_k и фактическое \tilde{A}_k . Второе в случае счетчиков представляет собой пуассоновскую реализацию первого:

$$\tilde{A}_k \sim \mathcal{P}(A_k): P(\tilde{A}_k = a) = \exp\{-A_k\} A_k^a / a!, \quad a = 0, 1, \dots$$

Кроме интересующего нас сигнала A_k (*активного сигнала*) учтем еще существование фона B_k , который также будем считать пуассоновски распределенным независимо от активного:

$$\tilde{B}_k \sim \mathcal{P}(B_k): P(\tilde{B}_k = b) = \exp\{-B_k\} B_k^b / b!, \quad b = 0, 1, \dots$$

Несложно показать, что сумма активного и фонового сигнала, $c_k \equiv \tilde{A}_k + \tilde{B}_k$, также является пуассоновской случайной величиной, с соответствующим ожиданием (сумма пуассоновских величин – пуассоновская величина!):

$$c_k \sim \mathcal{P}(A_k + B_k): P(c_k = c) = \exp\{-(A_k + B_k)\} (A_k + B_k)^c / c!, \quad c = 0, 1, \dots$$

Внутри канала трансформируется именно суммарный сигнал c_k . Как же мы хотим разделить активный и фоновый сигналы, чтобы, основываясь на данных (выходном сигнале детектора), реконструировать входной сигнал A_k (или \tilde{A}_k в зависимости от постановки задачи)? Основная идея заключается в том, что активный сигнал и фон имеют какие-то разные свойства (например, степень гладкости или наличие/отсутствие высокочастотных компонент) – эту информацию в байесовскую модель можно передать в виде информативного прайора! Мы еще вернемся к этому вопросу позже, когда будем обсуждать априорные распределения.

Предположим, что трансформация сигнала $C_k = f_k(c)$ осуществляется по некоторому известному для нас закону с известным же параметром(ами) ϵ . Например, в текущей схеме Виктора используется преобразование, которое после масштабирования может быть представлено в виде:

$$C_k = f_k(c) = c_k + \epsilon c_{k-1} + \epsilon^2 c_{k-2} + \dots + \epsilon^{k-1} c_1 \quad (16)$$

Наконец измеряемые на выходе данные сами «страдают» пуассоновскими флуктуациями, т.е.

$$d_k \sim \mathcal{P}(C_k): P(d_k = d) = \exp\{-C_k\} C_k^d / d!, \quad d = 0, 1, \dots$$

Байесовская постановка задачи заключается в вычислении постериорного распределения

$$p(A, B | \mathcal{D}) \propto p(\mathcal{D} | A, B) p(A) p(B), \quad A = \{A_k\}, B = \{B_k\}, \mathcal{D} = \{d_k\},$$

где мы дополнительно предположили априорную независимость активного и фонового сигналов.

9.1.1 Функция правдоподобия

Сначала подробнее рассмотрим правдоподобие $p(\mathcal{D}|A, B)$. Его можно представить посредством маргинализации по скрытым переменным $c = \{c_k\}$ и ...

$$p(\mathcal{D}|A, B) = \sum_c p(\mathcal{D}|c) p(c|A + B) = \sum_c \int \prod_{k=1}^K dC_k p(d_k|C_k) p(c_k|A_k + B_k) p(\{C_k\}|\{c_k\})$$

Последний множитель в этой формуле представляет собой просто дельта-функцию, поэтому многократный интеграл просто означает, что везде вместо C_k требуется подставить его выражение через c из (16). Обращаем внимание, что $C_k(c) = f_k(c)$ зависит только от всех предыдущих значений c (c_k, c_{k-1}, \dots, c_1), но не последующих. При желании это преобразование можно представить в матричном виде $C = \mathcal{E}c$ с ниже-треугольной матрицей \mathcal{E} (у нее на диагонали стоят 1, а внедиагональные элементы равны степеням ϵ).

Таким образом,

$$p(\mathcal{D}|A, B) = \sum_c \prod_{k=1}^K p(d_k|f_k(c)) p(c_k|A_k + B_k)$$

Конечно же, эта формула для правдоподобия справедлива и в более общем случае непуассоновских статистик (не пуассоновскими могут быть как ошибки измерений d_k , так и флуктуации входного сигнала). К сожалению, даже в случае пуассоновских распределений сумму по всем значениям вектора c в явном виде взять нельзя (точно ли?) из-за наличия недиагональных членов в матрице преобразования \mathcal{E} .

Зато такая функция правдоподобия легко строится средствами вероятностного программирования в РумС.

9.1.2 Прайоры

Выше мы уже отмечали, что правдоподобие зависит лишь от суммы активного и фонового сигнала. Поэтому разделить информацию от этих двух составляющих (и, как следствие, реконструировать интересующий нас активный сигнал) можно лишь при наличии информативного прайора.

Самый простой способ сделать это – искать фон и активный сигнал в разных семействах функций. Например, фоновый сигнал B представлять полиномами низкого порядка (0 – постоянный фон, 1 – линейный, 2 – квадратичный), тогда как сигнал может иметь импульсный характер. Рассмотрим для определенности ситуацию с линейной моделью фона и одиночным прямоугольным импульсом:

$$B_k = B^{\text{lin}}(t_k) = B_0 + B_1 t_k, \quad A_k = A^{\text{imp}}(t_k) = A_0 \quad \text{iff } t \in [t_{\text{st}}, t_{\text{st}} + \Delta t_{\text{imp}}]$$

В этом случае мы имеем дело с 5-параметрической задачей, $\Theta = \{B_0, B_1, A_0, \Delta t_{\text{st}}, t_{\text{imp}}\}$. Прайоры на эти параметры можно задавать факторизованными (независимыми) и малоинформативными – достаточно только учесть положительность B_0, A_0 и границы значений временных параметров. Разделение фонового и активного сигнала в процессе байесовского вывода произойдет автоматически на основании резких (однотактовых) скачков сигнала.

Такая модель легко обобщается на случай активного сигнала в виде серии прямоугольных импульсов. Однако при реализации в РумС надо каким-то образом задать упорядоченное множество времени начала импульсов и отследить наличие ненулевых промежутков между импульсами (иначе это уже несколько другая задача!).

9.1.3 ВАЖНЫЕ ВОПРОСЫ

Изначально схема $C = \mathcal{E}c$ выбиралась с целью максимального упрощения процедуры обращения – восстановления сигнала по измерениям (идея была восстанавливать активный сигнал по двум последним отсчетам). После того, как мы начали решать задачу восстановления байесовскими методами, может возникнуть вопрос: а не существует ли какой-то оптимальной схемы, т.е. такой измерительной схемы $C = f(c)$, которая позволит восстанавливать сигнал A_k с наименьшей ошибкой? Естественно, при этом имеет смысл рассматривать лишь такие схемы, которые могут быть реализованы аппаратным образом.

Еще один вопрос, имеющий важное практическое значение, а нельзя ли сформировать он-лайн версию приведенной выше модели? То есть сделать так, чтобы восстанавливать значение сигнала в каждый текущий момент времени. При этом конечно, должна происходить и пере-оценка предыдущих значений. Возможна ли формулировка такой байесовской модели, в которой данные поступают порционно?

В свете предыдущих двух вопросов возникает желание «увидеть» в нашей модели хорошо известную в науке *модель пространства состояний* SSM (state-space model), для которой как раз и разрабатываются эффективные он-лайн методики восстановления состояний системы. В принципе, если состоянием в нашем случае назвать совокупность переменных $A_k, B_k, \tilde{A}_k, \tilde{B}_k, C_k$, то мы и получим максимально близкую к SSM формулировку. Однако ввиду наличия вероятностных зависимостей между A_k в различные моменты времени, $p(A) \neq p(A_1)p(A_2) \dots p(A_K)$ (и аналогично для фона), такой подход отличается от SSM.

9.2 Фильтры Калмана

Еще одним, пожалуй, наиболее оптимистичным подходом для решения данной задачи является так называемый *фильтр Калмана*. Довольно просто про этот фильтр рассказывается вот в этой заметке на Хабре: [Объяснение фильтра Калмана в картинках](#). Также рекомендую прочесть научно-популярную статью, на основании которой и написана эта заметка: [Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation](#). На официальном сайте MATWORKS есть хорошо понятная документация по этому фильтру, см. [Kalman Filtering](#).

Наконец, есть очень детальный разбор (почти) всех нюансов фильтра Калмана в монографии М.Н.Grewal and А.Р.Andrews, Kalman Filtering: Theory and Practice Using MATLAB. Третье издание книги (2008) доступно по [ссылке](#), а matlab-коды к примерам четвертого издания см. [здесь](#). Также можно порекомендовать [Github-репозиторий](#) Wenbo Chen Chaphlagical, посвященный данной книге.

Примечание. Возможно, наиболее полезным для понимания вероятностной подоплеку фильтров Калмана станет прочтение соответствующих глав в книгах Мерфи (в наших обозначениях [Book0](#), [Book1](#) и [Book2](#)). Речь, конечно же, идет о разделах, посвященных posterior inference in state-space models. В книге Book0 (2012 года) в первую очередь рекомендую посмотреть главы 17 (Markov and hidden Markov models) и 18 (State-space models). Аналогичные вопросы обсуждаются также в Book2 – см. главы 8 (Gaussian filtering and smoothing) и 29 (State-space models). Главное отличие (помимо того, что в новой версии книги главы, посвященные дискретным и гауссовым моделям, поменялись местами) заключается в том, что весь код Book0 написан на Matlab (и доступен по адресу <https://github.com/probml/pmtk3>), а в Book2 – на Python (см. <https://github.com/probml/probml>).

Наконец, наиболее детальный разбор вопросов, связанных с различными вариантами обобщения фильтров Калмана (в рамках Байесового подхода) проведен в *Bayesian-Filtering-and-Smoothing*, Simo Sarkka and Lennart Svensson (она есть у нас в директории *ProbProgramming/Books/*). Большинство примеров из этой книги доступны как в Matlab, так и Python-коде: <https://github.com/EEA-sensors/Bayesian-Filtering-and-Smoothing>.

9.3 Time Series in PyMC

Пример [Analysis of an Model in PyMC](#) может служить кратким введением как работать с авторегрессионными моделями. Если же на повестке дня стоит вопрос байесовского прогноза в рамках таких моделей, то следует обратиться к примеру [Forecasting with Structural AR Timeseries](#). В этом примере модель обучает авторегрессионную модель с трендом и сезонным эффектом, а затем выполняет предсказание в будущее.

Гораздо более общий подход предложенным в довольно сложном примере [Time Series Models Derived From a Generative Graph](#): здесь не используются предопределенные модели стохастических временных рядов – они конструируются с помощью генеративного графа. В этом случае для циклического выполнения обновлений (шагов) внутри модели вызывается функция `pytensor.scan()`.

10 Задача Романа Сараева

Данный раздел посвящен различным формулировкам байесовских моделей так называемых 3D-трековых событий, т.е. таких движущихся в пространстве светящихся объектов, которые при проецировании оптикой детектора на фокальную поверхность (ФП) оставляет след в виде трека. Кинематика изображения (track event, TE) определяется кинематикой источника (3D-TE). В зависимости от величины скорости объекта v имеет смысл различать три ситуации: 1) нерелятивистское событие (N-TE), $v \ll c$ ⁸, 2) релятивистское событие (R-TE), $v \sim c$ и 3) ультра-релятивистское событие (U-TE), $v = c$. Несложно показать, что в наиболее общем виде (вариант 2) изменение луча зрения \vec{e} (единичного вектора от детектора D до источника S) может быть охарактеризовано как

$$\omega r = \frac{v \sin \chi}{1 + \frac{v}{c} \cos \chi}$$

где $r \equiv DS$ – расстояние от детектора до источника, а χ – угол между \vec{e} и \vec{v} .

В случаях N-TE и U-TE это соотношение упрощается до

$$\omega r/v = \sin \chi \quad \text{и} \quad \omega r/c = \tan(\chi/2)$$

соответственно.

Введем сферическую систему координат с углами γ, ψ для вектора \vec{e} и θ, ϕ для вектора $-\vec{v}/v$ (т.е. вектора, противоположного направлению движения источника⁹):

$$\vec{e} \equiv \vec{r}/r = (\sin \gamma \cos \psi, \sin \gamma \sin \psi, -\cos \gamma), \quad \vec{s} \equiv \vec{v}/v = (-\sin \theta \cos \phi, -\sin \theta \sin \phi, -\cos \theta)$$

Тогда $\cos \chi = (\vec{e} \cdot \vec{s}) = \cos \gamma \cos \theta - \sin \gamma \sin \theta \cos \delta\phi$, где $\delta\phi = \phi - \psi$.

Обратное преобразование, позволяющее по измерению χ определить θ , можно представить в виде

$$\cos(\theta + \delta\theta) = \cos \chi \frac{\cos \delta\theta}{\cos \gamma} = \frac{\cos \chi}{\sqrt{\cos^2 \gamma + \sin^2 \gamma \cos^2 \delta\phi}}, \quad \tan \delta\theta \equiv \tan \gamma \cos \delta\phi$$

При наблюдении с орбиты в надир $\gamma = 0$ и $\chi = \theta$. Для источника, расположенного недалеко от оси (что зачастую справедливо в случае узкоугольной оптики), отличие χ от θ может рассматриваться в качестве малого параметра и тогда

$$\chi \approx \theta + \gamma \cos \delta\phi - \frac{\gamma^2 \sin^2 \delta\phi}{2 \tan \theta}, \quad \theta \approx \chi - \gamma \cos \delta\phi + \frac{\gamma^2 \sin^2 \delta\phi}{2 \tan \chi} \quad (17)$$

Примечание. Первое из уравнений (17) нельзя использовать при $\theta \rightarrow 0$. Это связано с тем, что в этом случае у нас сразу два малых параметра, γ и θ , и предельный переход следует выполнять аккуратнее. Можно показать, что правильным выражение для предельного перехода будет $\chi = \sqrt{\theta^2 + 2\theta\gamma \cos \delta\phi + \gamma^2}$. В пределе $\theta \rightarrow \pi$ аналогичная формула выглядит так: $\pi - \chi = \sqrt{\epsilon^2 - 2\epsilon\gamma \cos \delta\phi + \gamma^2}$, где $\epsilon \equiv \pi - \theta \rightarrow 0$.

В приборе событие представляет собой трек $\mathbf{R}(T)$, по которому движется трековая точка (центроид изображения) со скоростью \mathbf{U}^{10} . Можно показать, что в случае N-TE и

⁸Здесь $c \approx 0.3 \text{ km}/\mu\text{s}$ – скорость света в вакууме.

⁹Такой необычный выбор удобен при рассмотрении наиболее важной для нас задачи – детектирования ШАЛ с околоземной орбиты, когда этими углами характеризуют направление прилета частицы КЛ ПВЭ.

¹⁰Здесь и далее выделенные жирным символы представляют двумерные векторы, заданные на ФП прибора.

если оптика прибора является идеальным проектором на ФП с фокусным расстоянием f , то

$$f\vec{\omega} = U \frac{\vec{n} + \vec{e}'_z(R/f) \sin \Delta\Phi}{1 + R^2/f^2},$$

где U, R – модули векторов \mathbf{U}, \mathbf{R} (\mathbf{R} отсчитываем из центра ФП), \vec{e}'_z – единичный вектор в направлении главной оптической оси, $\vec{n} \equiv [\vec{e}'_z \times \vec{U}]/U$ (т.е. \vec{n} – единичный вектор, лежащий в ФП и перпендикулярный \mathbf{U}), а $\Delta\Phi$ – угол между векторами \mathbf{U} и \mathbf{R} , $\Delta\Phi = \Phi - \Psi$, $\tan \Phi = U_Y/U_X$, $\tan \Psi = Y/X$. Отсюда

$$\omega = \frac{U}{f} \frac{\sqrt{1 + (R/f)^2 \sin^2 \Delta\Phi}}{1 + (R/f)^2} \quad (18)$$

Таким образом, в случае N-ТЕ измерение векторов трековой скорости и положения объекта на ФП при известном расстоянии r позволяет определить направление прихода объекта. Более общая ситуация рассмотрена ниже.

10.1 Общая постановка задачи

Рассмотрим общую постановку задачи для движения объекта с постоянной скоростью \vec{v} , когда радиус-вектор \vec{r} «детектор–источник» может быть представлен в виде

$$\vec{r} = \vec{r}_0 + \vec{v}t = r_0\vec{e}_0 + vt\vec{s}$$

При этом учтем, что «детекторное» время T отличается от времени в источнике t ¹¹:

$$T = t + r - r_0$$

Таким образом, функция $\vec{r}(T)$ задана неявным образом (параметрически – посредством параметра t) и с точки зрения построения байесовой модели это не очень удобно.

Общее число параметров в такой кинематической модели равно 6 (три у \vec{r}_0 и три у \vec{v}), в случае U-NE оно уменьшается до 5 (так как $v = c$).

Оказывается в случае U-TE легко получить эту функцию в явном виде (случай N-TE подробно разобран в конце данного раздела). Действительно, при $v = 1$ имеем

$$\vec{r}/r_0 = \vec{e}_0 + \vec{s} \frac{\zeta + \zeta^2/2}{1 + \cos \chi_0 + \zeta}, \quad \zeta \equiv T/r_0, \quad \cos \chi_0 \equiv (\vec{e}_0 \cdot \vec{s}) \quad (19)$$

$$r/r_0 = 1 + \frac{\zeta \cos \chi_0 + \zeta^2/2}{1 + \cos \chi_0 + \zeta} = \frac{(1 + \cos \chi_0)(1 + \zeta) + \zeta^2/2}{1 + \cos \chi_0 + \zeta}$$

По определению $\vec{\omega} = [\vec{r} \times d\vec{r}/dT]/r^2$, и несложно показать, что

$$\vec{\omega}r = [\vec{r} \times d\vec{r}/dT]/r = \frac{[\vec{e}_0 \times \vec{s}]}{1 + \cos \chi_0 + \zeta},$$

причем *SSH: Проверить!*

$$\omega r = \frac{\sin \chi_0}{1 + \cos \chi_0 + \zeta} = \frac{\sin \chi}{1 + \cos \chi} = \tan(\chi/2)$$

¹¹Для удобства с этого момента будем полагать $c \equiv 1$. Обращаем внимание, что фигурирующие здесь и далее t и T на самом деле являются интервалами времени Δt и ΔT между соответствующими событиями (т.е. излучением и приемом сигнала) для \vec{r}_0 и \vec{r} .

Наконец, для динамики единичного вектора $\vec{e} \equiv \vec{r}/r$ (луча зрения) имеем

$$\vec{e}(\zeta) = \frac{(1 + \cos \chi_0 + \zeta)\vec{e}_0 + (\zeta + \zeta^2/2)\vec{s}}{(1 + \cos \chi_0)(1 + \zeta) + \zeta^2/2} \quad (20)$$

Именно эта зависимость определяет кинематику U-ТЕ на ФП, которая получается в результате преобразования \vec{e} оптической системой. В простейшем случае это преобразование заключается в комбинации поворота \mathcal{O}_d (в систему координат детектора) и проецирования \mathcal{P}_f :

$$\mathbf{R}_c(\zeta) = \mathcal{P}_f[\mathcal{O}_d[\vec{e}(\zeta)]],$$

где \mathbf{R}_c – двумерный вектор положения центра изображения на ФП.

Примечание. Типичная длительность события типа ШАЛ 100 μs , поэтому при детектировании с орбиты МКС $\zeta = T/r_0 < 100 \cdot 0.3/420 = 0.07$ и

$$\vec{e}(\zeta) \propto (1 + \cos \chi_0)\vec{e}_0 + \zeta(\vec{e}_0 + \vec{s}) + \vec{s}\zeta^2/2,$$

где последним (квадратичным) слагаемым в первом приближении можно пренебречь. В этом же приближении

$$r/r_0 \approx 1 + \frac{\zeta \cos \chi_0}{1 + \cos \chi_0}, \quad \omega \approx \frac{\tan(\chi_0/2)}{r_0} \left(1 - \zeta \frac{1 + 2 \cos \chi_0}{(1 + \cos \chi_0)^2} \right)$$

Обратимся теперь к общему случаю R-ТЕ, когда $v < 1$. При этом для выражения $r(T)$ придется решать квадратное уравнение с дискриминантом

$$\mathcal{D} = (1 + vc_0)^2 + 2\zeta(v + c_0) + \zeta^2 = (1 - v^2)(1 - c_0^2) + (v + c_0 + \zeta)^2,$$

где определение ζ немного модифицировалось, $\zeta \equiv vT/r_0$, и мы ввели короткое обозначение $c_0 \equiv \cos \chi_0$. Второе выражение (эквивалентное первому) удобно для получения предельного перехода при $v \rightarrow 1$ (см. ниже).

В результате получим

$$r/r_0 = 1 + \frac{\sqrt{\mathcal{D}} - (1 + vc_0 + v\zeta)}{1 - v^2}, \quad \zeta \equiv vT/r_0$$

$$\vec{r}/r_0 = \vec{e}_0 + \vec{s}(\zeta - v(r/r_0 - 1)), \quad \vec{e} = \frac{\vec{r}/r_0}{r/r_0}$$

Рассмотрим отдельно переход к случаю $v \rightarrow 1$, введя малый параметр $\epsilon = 1 - v$. Тогда

$$\mathcal{D} = (1 + c_0 + \zeta)^2 - 2\epsilon(c_0 + \zeta + c_0^2) + \epsilon^2 c^2,$$

$$\sqrt{\mathcal{D}} \approx 1 + c_0 + \zeta - \epsilon \frac{c_0 + \zeta + c^2}{1 + c_0 + \zeta} - \frac{\epsilon^2 (c_0 + \zeta + c_0^2)^2 - c_0^2(1 + c_0 + \zeta)^2}{2(1 + c_0 + \zeta)^3}$$

$$\frac{r}{r_0} - 1 = \zeta \frac{c_0 + \zeta/2}{1 + c_0 + \zeta} \left\{ 1 + \epsilon \frac{c_0^2 + (c_0 + \zeta)^2 + 2(c_0 + \zeta)}{(1 + c_0 + \zeta)^2} \right\}$$

$$\vec{r}/r_0 = \vec{e}_0 + \vec{s} \frac{\zeta}{1 + c_0 + \zeta} \left\{ 1 + \zeta/2 + \epsilon(c_0 + \zeta/2) \frac{1 - c_0^2}{(1 + c_0 + \zeta)^2} \right\}$$

Примечание. Можно показать, что движение изображающей точки по ФП в случае идеального проектирования определяется в U-TE выражением (для простоты полагаем $f = 1$ и не учитываем матрицу поворота \mathcal{O}_d):

$$\mathbf{R} = \mathbf{R}_0 + \frac{\mathcal{A} \cos \theta}{\mathcal{A} \cos \theta + \cos \gamma_0} (\mathbf{s} \tan \theta - \mathbf{R}_0) = \mathbf{R}_0 + \left(1 - \frac{1}{1 + \frac{\mathcal{A} \cos \theta}{\cos \gamma_0}} \right) (\mathbf{s} \tan \theta - \mathbf{e}_0 \tan \gamma_0) \quad (21)$$

причем (возвращая обозначение c для скорости света)

$$\mathcal{A} \equiv \frac{\zeta + \zeta^2/2}{1 + \cos \chi_0 + \zeta} = \frac{\zeta}{2} \left[1 + \frac{1 - \cos \chi_0}{1 + \cos \chi_0 + \zeta} \right], \quad \zeta \equiv cT/r_0$$

где $\tan \gamma_0 = |\mathbf{R}_0|$, $\mathbf{e}_0 \equiv \mathbf{R}_0/|\mathbf{R}_0|$, $\cos \chi_0 = \cos \theta \cos \gamma_0 (1 + (\mathbf{e}_0 \cdot \mathbf{s}) \tan \theta \tan \gamma_0)$. Здесь единичный двумерный вектор \mathbf{s} составлен из компонент вектора направления движения источника \vec{s} , $\mathbf{s} \propto (s_x, s_y)$. Обращаем внимание, что движение по ФП происходит не вдоль \mathbf{s} , а вдоль линейной комбинации $\mathbf{s} \tan \theta - \mathbf{e}_0 \tan \gamma_0$.

Несмотря на довольно сложный вид этого выражения, кинематика движения изображающей точки определяется всего пятью параметрами: θ , γ_0 , ориентацией двумерных векторов \mathbf{e}_0 и \mathbf{s} , а также нормирующим параметром r_0 , входящим в определение $\zeta (\equiv T/r_0)$.

Движение изображающей точки удобно параметризовать посредством начального положения \mathbf{R}_0 и начальной скорости $\mathbf{U}_0 \equiv d\mathbf{R}(T)/dT|_{T=0}$. Можно показать, что

$$\mathbf{s} \tan \theta - \mathbf{e}_0 \tan \gamma_0 = r_0 (1 + \cos \chi_0) \frac{\cos \gamma_0}{\cos \theta} \mathbf{U}_0$$

Это позволяет выразить движение изображающей точки через \mathbf{R}_0 , \mathbf{U}_0 и r_0 (снова всего 5 параметров). В частности,

$$\frac{1 + \cos \chi_0}{2} = \frac{1}{1 - r_0^2 \sin^2 \gamma_0 \cos^2 \gamma_0 (\mathbf{e}_0 \cdot \mathbf{U}_0)^2 + r_0^2 \mathbf{U}_0^2 \cos^2 \gamma_0} = \frac{(1 + \rho_0^2)^2}{(1 + \rho_0^2)^2 - u_0^2 \rho_0^2 \cos^2 \delta_0 + u_0^2}$$

$$\frac{\cos \theta}{\cos \gamma_0} = \frac{[1 - r_0 \sin \gamma_0 \cos \gamma_0 (\mathbf{e}_0 \cdot \mathbf{U}_0)]^2 - r_0^2 \mathbf{U}_0^2 \cos^2 \gamma_0}{1 - r_0^2 \sin^2 \gamma_0 \cos^2 \gamma_0 (\mathbf{e}_0 \cdot \mathbf{U}_0)^2 + r_0^2 \mathbf{U}_0^2 \cos^2 \gamma_0} = \frac{[1 + \rho_0^2 - \rho_0 u_0 \cos \delta_0]^2 - u_0^2}{(1 + \rho_0^2)^2 - u_0^2 \rho_0^2 \cos^2 \delta_0 + u_0^2}$$

где мы для удобства ввели (безразмерные) модули векторов \mathbf{R}_0 и \mathbf{U}_0 , $\rho_0 \equiv \tan \gamma_0 = |\mathbf{R}_0|/f$, $u_0 = r_0 |\mathbf{U}_0|/(fc)$ и угол между ними, $\cos \delta_0 = (\mathbf{R}_0 \cdot \mathbf{U}_0)/(|\mathbf{R}_0| |\mathbf{U}_0|)$.

Подставляя эти выражения в

$$\mathbf{R} - \mathbf{R}_0 = \left(1 - \frac{1}{1 + \frac{\mathcal{A} \cos \theta}{\cos \gamma_0}} \right) (1 + \cos \chi_0) \mathbf{U}_0 = \frac{(1 + \cos \chi_0) \mathbf{U}_0}{1 + \frac{\cos \gamma_0}{\cos \theta} \frac{1}{\mathcal{A}}} \quad (22)$$

получаем

$$\mathbf{R} - \mathbf{R}_0 = \frac{2C \mathbf{U}_0 r_0 / c}{1 + \frac{2}{[1 + \rho_0^2 - \rho_0 u_0 \cos \delta_0]^2 - u_0^2} \frac{1}{\zeta} \frac{C + \zeta/2}{1 + \zeta/2}}, \quad C = \frac{1 + \cos \chi_0}{2}$$

Таким образом, при разложении зависимости $\mathbf{R}(T)$ вблизи $T = 0$ достаточно будет задать начальное положение \mathbf{R}_0 , начальную скорость \mathbf{U}_0 и модуль начального ускорения A_0 (направление ускорения совпадает с направлением \mathbf{U}_0). К сожалению явный вид выражений для \mathbf{U}_0 и A_0 довольно сложный, что не позволяет выразить через них исходные параметры кинематики U-TE. *SSh: Точно? Он ведь понадобится для того, чтобы по равноускоренной модели восстанавливать параметры источника!*

Выражение (21) справедливо и в общем случае R-TE, только надо поменять определение величин \mathcal{A} и ζ :

$$\mathcal{A} = \zeta - v \frac{\sqrt{D} - (1 + vc_0 + v\zeta)}{1 - v^2}, \quad \zeta = vT/r_0$$

В этом случае число независимых параметров увеличится до шести – добавится модуль скорости движения источника v . *SSh: А как быть с разложением по T ? Так как по-прежнему ускорение направлено вдоль скорости, то в разложении до квадратичных членов (равноускоренное движение) шесть параметров «свернутся» лишь в пять комбинаций.*

Наконец, в нерелятивистском случае N-RE кинематика изображающей точки максимально проста:

$$\mathbf{R} = \mathbf{R}_0 + \frac{\mathbf{U}_0 T}{1 + \nu T}, \quad \mathbf{U}_0 = \nu(\mathbf{s} \tan \theta - \mathbf{R}_0), \quad \nu \equiv \frac{v \cos \theta}{r_0 \cos \gamma_0}$$

причем $\mathbf{U}(T) \equiv d\mathbf{R}(T)/dT = \mathbf{U}_0/(1 + \nu T)^2$. Обращаем внимание, что число параметров, однозначно определяющих движение изображающей точки, равно 5, что свидетельствует о неполноте данных с точки зрения реконструкции кинематической модели движения источника (6 параметров).

Обобщим это выражение на случай нерелятивистского источника, движущегося прямолинейно и равноускоренно, т.е. когда модуль его скорости меняется¹² как $v + aT$:

$$\mathbf{R} = \mathbf{R}_0 + \frac{\mathbf{U}_0 T(1 + \alpha T)}{1 + \nu T(1 + \alpha T)}, \quad \mathbf{U}_0 = \nu(\mathbf{s} \tan \theta - \mathbf{R}_0), \quad \nu \equiv \frac{v \cos \theta}{r_0 \cos \gamma_0}, \quad \alpha \equiv \frac{a}{2v}$$

При квадратичном разложении $\mathbf{R}(T) \approx \mathbf{R}_0 + \mathbf{U}_0 T + \mathbf{A}_0 T^2/2$ получим $\mathbf{A}_0 = 2(\alpha - \nu)\mathbf{U}_0$. В виду коллинеарности \mathbf{U}_0 и \mathbf{A}_0 число параметров такой аппроксимации равно 5 и не позволяет воостановить все 7 параметров исходной кинематической модели равноускоренного движения источника. В исходной параметризации движения изображающей точки число параметров равно 6, что тоже не позволяет выполнить полную реконструкцию кинематики источника.

Таким образом, только в U-TE движение источника можно реконструировать основываясь на кинематических данных по изображающей точке («тайминг»). Если же скорость движения источника неизвестна (а тем более, когда движение источника происходит с ускорением), то для реконструкции требуется дополнительная информация, заданная детерминированной связью или вероятностным распределением. Это может быть известная высота полета (при движении спутника), вероятное направление прилета (в случае потоковых метеоров), грубая оценка расстояния до объекта r_0 и др. В некоторых случаях при формулировке вероятностной модели придется фигурирующий в связи параметр(ы) ввести в параметризацию модели.

Примечание. Выше было рассмотрено представление в системе координат, связанной с детектором (СКД). При решении практических задач это не всегда удобно. Поэтому выразим фигурировавшие кинематические характеристики при постановке задачи в произвольной СК (назовем ее *глобальной*, ГСК). В этой системе движение источника света описывается как $\vec{R}_s(t) = \vec{R}_0 + \vec{v}t$, а детектор покоится $\vec{R}_d = \text{const}$. (В случае релятивистских событий смещением любого типа детектора за время регистрации можно пренебречь,

¹²Нерелятивистские источники «гораздо охотнее» изменяют свою скорость чем релятивистские – например, метеоры тормозятся в атмосфере.

а при детектировании нерелятивистских объектов с орбиты, например метеоров, достаточно будет подставить в $\vec{R}_s(t)$ в качестве скорости относительную скорость, $\vec{v} = \vec{v}_s - \vec{v}_d$.)

Тогда для N-ТЕ имеем

$$\nu \equiv \frac{\vec{v} \cdot \vec{d}}{\vec{r}_0 \cdot \vec{d}} \quad \vec{r}_0 \equiv \vec{R}_d - \vec{R}_0,$$

где \vec{d} – единичный вектор, определяющий направление главной оптической оси детектора (перпендикулярно этому направлению лежит плоскость проецирования оптики).

С параметрами \mathbf{R}_0 и \mathbf{U}_0 чуть сложнее, так как их выражение требует введение матрицы \mathcal{O}_d , осуществляющей преобразование в СКД:

$$\mathbf{R}_0/f = \frac{\mathcal{O}_d[\vec{r}_{0\perp}]}{\vec{r}_0 \cdot \vec{d}}, \quad \mathbf{U}_0/f = \frac{\mathcal{O}_d[\vec{v}_\perp]}{\vec{r}_0 \cdot \vec{d}} - \nu \mathbf{R}_0/f = \frac{\mathcal{O}_d[\vec{v} - \nu \vec{r}_0]}{\vec{r}_0 \cdot \vec{d}},$$

где $\vec{r}_{0\perp} \equiv \vec{r}_0 - (\vec{r}_0 \cdot \vec{d})\vec{d}$, $\vec{v}_\perp \equiv \vec{v} - (\vec{v} \cdot \vec{d})\vec{d}$.

10.2 Байесовская модель стерео-метеора

При заданном направлении прилета метеора его трековое изображение инвариантно при одновременном масштабировании скорости метеороида и радиус-вектора детектор-метеороид. Поэтому если сформулировать байесовскую модель в терминах не трекового события на ФП, а как 3D-модель движущегося метеороида, кинематика которого в простейшем случае задается его положением в некоторый момент времени \vec{r}_0 и скоростью \vec{v} , то даже при большом количестве данных постериорное распределение будет во многом определяться прайорами (так как правдоподобие является лишь функцией отношения v/r_0). Однако байесовская модель реконструкции может стать весьма эффективной в ситуации стерео-регистрации события двумя или большим количеством детекторов.

Рассмотрим случай стерео-метеора, регистрация свечения которого ведется одновременно с помощью двух детекторов изображения и набор данных состоит из сигналов в N_1 и N_2 каналах: $\mathcal{D} \equiv \{\mathcal{D}_{(1)}, \mathcal{D}_{(2)}\} = \{S_{(1)ik}, S_{(2)jk}\}$, $i = 1, \dots, N_1$, $j = 1, \dots, N_2$ (для простоты будем считать, что такты измерения t_k совпадают, $k = 1, \dots, K$).

Правдоподобие для стерео-данных может быть факторизовано как

$$p(\mathcal{D}|\Theta_{\text{kin}}, \Theta_{\text{dyn}}) \propto \prod_{k=1}^K p(\mathcal{D}_1|\mathbf{R}_{(1)k}, I_{(1)k}) p(\mathcal{D}_2|\mathbf{R}_{(2)k}, I_{(2)k}), \quad (23)$$

где $\mathbf{R}_{(n)k} = \mathcal{O}_{(n)}[\vec{r}_0 + \vec{v}t_k - \vec{r}_{(n)d}]$ – вектор центра изображения метеора на ФП детектора номер n (оператор $\mathcal{O}_{(n)}$ – композиция матрицы вращения, задаваемой ориентацией детектора, и проецирования), а $\vec{r}_{(n)d}$ – (известный) вектор положения самого детектора. Интенсивность свечения сигнала может быть параметризована одной и той же функцией (с параметрами Θ_{dyn})

$$I_{(n)k} = f(t_k|\Theta_{1c}) \cdot \eta(|\vec{r}_0 + \vec{v}t_k - \vec{r}_{(n)d}|),$$

причем амплитудный фактор η , учитывающий уменьшение интенсивности с расстоянием и ввиду рассеяния в атмосфере, в первом приближении может считаться не зависящим от времени, $\eta \approx \eta(|\vec{r}_0 - \vec{r}_{(n)d}|)$.

Каждый из факторов в (23) по-прежнему может быть выражен через функцию $\text{PED}_i(\mathbf{R})$, в простейшем случае как гауссова модель ошибок измерения (??).

Достоинством байесовского подхода является возможность учета априорной информации на параметры реконструкции \vec{v} ($\equiv v\vec{n}_v$), \vec{r}_0 ($\equiv r_0\vec{n}_r$). Так, если речь идет о потоковом метеоре, то,

$$p(\vec{v}, \vec{r}_0) = p(v|v_{\text{sh}}, \sigma_v) p(\vec{n}_v|\vec{n}_{\text{sh}}, \sigma_{\text{sh}}) p(H_0|H_{\text{in}}, \sigma_H) p(\vec{n}_r),$$

где первые два фактора учитывают характерные значения скорости потока и его радианта, а также их дисперсию, а третий может быть использован для указания интервала высот, на которых вероятно начало интенсивного свечения (оно может меняться в зависимости от происхождения потока). Последний фактор ограничен общим полем зрения детекторов.

Обобщение на случай равноускоренного движения или на одну из параметризованных феноменологических моделей замедления метеорода ??, и даже учет связи между кинематикой и динамикой метеора при таком подходе очевидно. При этом количество параметров модели (Θ_{kin} , Θ_{dyn} , вспомогательных параметров) значительно возрастает. В многопараметрическом пространстве постериорное распределение может оказаться многомодальным и широким, и роль физически выделенных прайоров как регуляризаторов значительно возрастает. Для унимодального и узкого распределения $p(\Theta_{\text{kin}}, \Theta_{\text{dyn}}|\mathcal{D})$ результаты реконструкции могут быть представлены в виде точечных оценок и корреляционных коэффициентов.

11 Введение в Эльфолокацию

В грозовой атмосфере происходят разнообразные процессы, многие из которых сопровождаются электрическими разрядами. Молния – самый известный из таких разрядов, но далеко не единственный. Помимо этой гигантской по размерам, току и излучению искры с помощью датчиков радиопульсов регистрируют и другие, которые зачастую ассоциируют с *механизмом инициации молнии*. Речь, в первую очередь, идет о NBEs (narrow bipolar events, *узкие биполярные события*) и PBs (preliminary breakdowns, *предварительных пробоях*), см. [KMS_2020].

NBE, который также называют CID (compact intracloud discharge, *компактный внутривлажный разряд*), в среднем локализован выше молниевых разрядов: так называемые положительные CIDs регистрируют на высотах 7-15 km (медиана 13 km), отрицательные CIDs – на высотах 15-20 km (медиана 18 km). Для CIDs характерно гигантское энерговыделение в VHF – в диапазоне 60-66 MHz их мощность может достигать 300 kW. Длительность типичного CID (на полуамплитуде VHF) составляет около 5 μ s (т.е. примерно 2 такта детектора «УФ атмосфера» и 6 тактов детектора «ТУС»).

Природа NBE на сегодняшний момент плохо понятна, да и сами эти события изучены недостаточно. Точность пространственной локализации разрядов, их размеры известны с большой неопределенностью, а информации об ориентации разряда практически нет совсем. NBEs и PBs регистрируют только по радиосигналу, так как выделение света в оптическом диапазоне от них незначительное.

Однако все виды мощных электрических разрядов в грозовой туче вызывают крупномасштабный отклик в ионосфере. Типичным «отпечатком» разряда, распростертым на сотни (и даже тысячу!) километров, является событие, получившее в литературе название ELVES (Emission of Light and Very low frequency perturbations due to Electromagnetic pulse sources), или просто «эльф».

Эльф образуется в результате «подогревания» ионосферы электромагнитным импульсом (EMP, electromagnetic pulse), сгенерированным разрядом в грозовой туче. На больших расстояниях от разряда фронт EMP можно считать приближенно сферическим и кольцеобразная структура области свечения эльфы соответствует пересечению этого фронта тонкого слоя ионосферы вблизи высоты 90 km.

11.1 Эльфы «УФ атмосферы»

Впервые изображение эльфы с околоземной орбиты было получено в ходе эксперимента ISUAL [Chern2003]. В [Marshall2008] эльфы наблюдали с поверхности в высокоскоростной (25 000 fps) фотометр PIPER. Этот детектор состоял лишь из двух линеек светочувствительных сенсоров и не позволял наблюдать всю кольцевую структуру свечения. Гораздо более интересным, с нашей точки зрения, оказались эльфы детектора «ТУС», проявляющие характерную пространственно-временную структуру свечения с временным разрешением 800 ns...

Подробный анализ эльфов PIPER приведен в работах [Newsom2010] и [Marshall2012].

Настоящей лабораторией по изучению эльфов стал орбитальный детектор «УФ атмосфера» – широкоугольный телескоп, работающий на борту МКС с 2019 года. Закрепленный с внутренней стороны МКС к УФ-прозрачному иллюминатору, этот телескоп наблюдает земную атмосферу в широком поле зрения (до 20° в диаметре с осью в надир). Размер проекции поля зрения на высоту свечения эльфы составляет примерно 250 × 250 km. Высокое временное разрешение (2.5 μ s) и большое количество УФ-чувствительных каналов

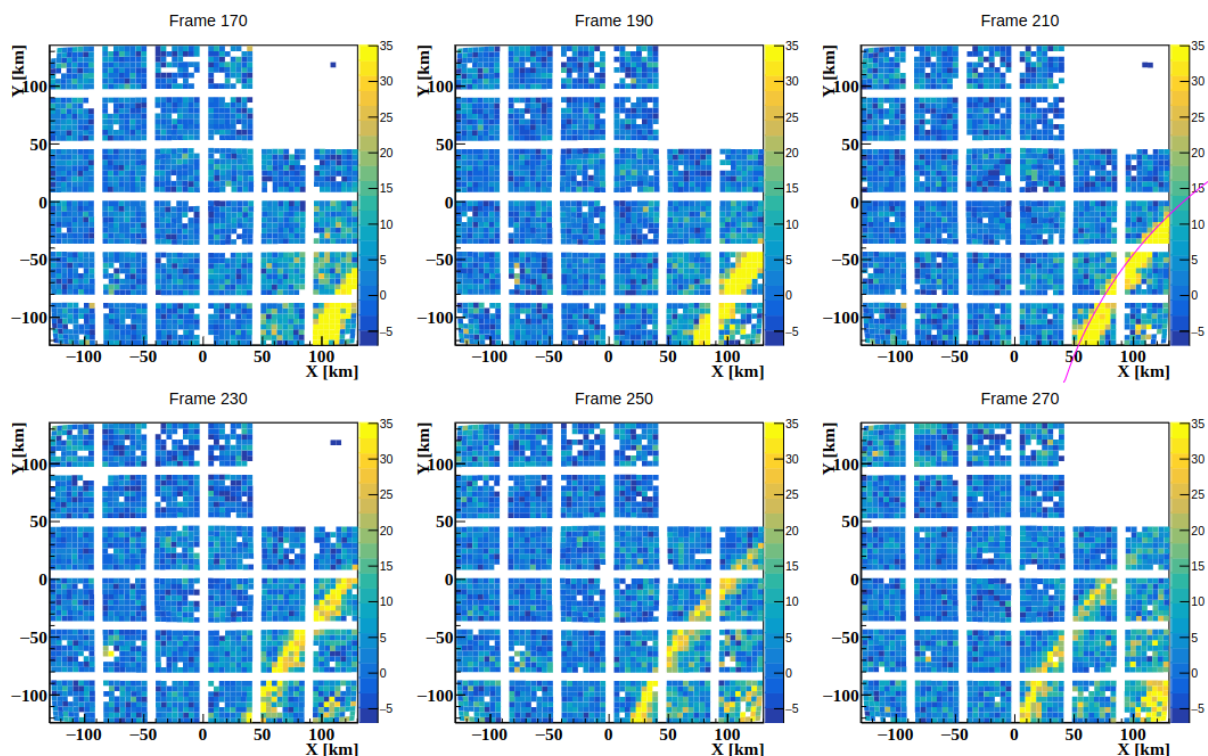


Рис. 2: Событие типа ELVES, зарегистрированное «УФ атмосфера» 5 декабря 2019 года. Показано шесть снимков с шагом $50 \mu\text{s}$ (20 тактов).

детектора (2304) позволяют детально отслеживать динамику эльфы на разных стадиях его развития, рис. 2. Серия кадров, представляющих собой мгновенные «изображения» (спроецированные на высоту 90 km) с шагом по времени $50 \mu\text{s}$, наглядно демонстрирует движения фронта свечения, а на последних двух кадрах хорошо также заметна неравномерность свечения кольца эльфы.

На рис. 3 представлены типичные сигналы события ELVES20191205, регистрируемые отдельными каналами детектора. На левой панели изображены сигналы в канале (34,13) и в 8 его «соседях» на фокальной поверхности (ФП) фотоприемника¹³. Во всех сигналах хорошо прослеживается резкий передний фронт роста свечения и чуть более медленный его спад. На правой панели два сигнала – каналов (37,13) и (36,14) – показаны в увеличенном виде, из которого хорошо заметна временная задержка порядка 10 тактов ($25 \mu\text{s}$), соответствующая перемещению фронта свечения на одну «диагональ» пикселя¹⁴.

11.2 Простейшая кинематическая модель эльфы

Рассмотрим простую модель эльфы, в которой земную атмосферу будем считать плоской, а в качестве данных фигурируют лишь времена пиков сигналов в каждом канале изображающего детектора, $\mathcal{D} = \{T_i\}$, здесь i – идентификатор канала, в котором удалось зарегистрировать «пик» эльфы. В таких данных «зашифровано» только положение разряда, но не его ориентация. Так что в качестве неизвестных параметров модели удобно выбрать координаты источника $S(x_0, y_0, z_0)$ – электрического разряда (который будем счи-

¹³Нумерация каналов ведется от 0 до 47 от левого нижнего угла ФП.

¹⁴Размер пикселя на высоте 90 km составляет примерно 5 km, поэтому смещению пика сигнала со скоростью $5\sqrt{2}/25 \approx 0.3 \text{ km}/\mu\text{s}$ свидетельствует об ультрарелятивистском перемещении фронта свечения.

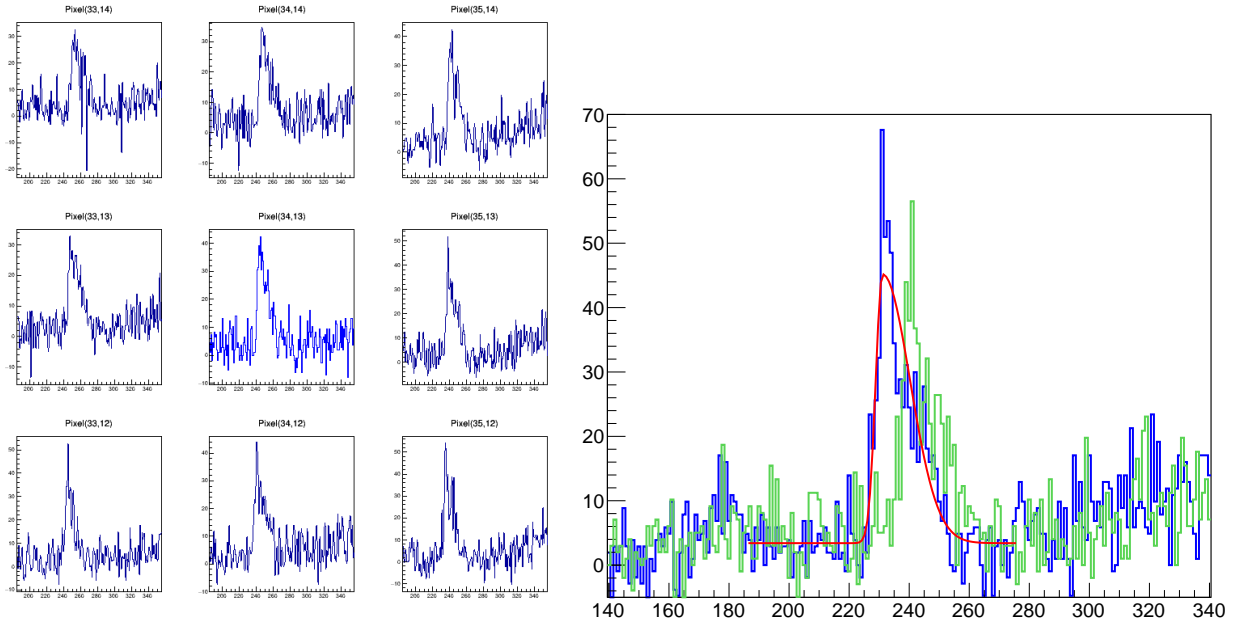


Рис. 3: Сигналы события ELVES20191205 в отдельных каналах. Слева – для 9 соседних каналов с центром в канале (34,13). Справа – сигналы каналов (37,13) (синий) и (36,14) (зеленый), соседних на фотоприемнике по диагонали. Красной линией проведен один из вариантов аппроксимации сигнала.

тать бесконечно малым) – в системе координат, связанной с надирной точкой детектора на поверхности Земли. Координаты детектора в этой СК $D(0, 0, z_D)$, где $z_D = H_{ISS} \approx 420$ km в случае детектора «УФ атмосфера» и $z_D = H_{TUS} \approx 500$ km в случае детектора «ТУС».

Наблюдаемый детектором эльф представляет собой совокупность светящихся «точек» E , расположенных на высоте z_E . Координаты каждой такой точки легко получить, если ввести единичный вектор вдоль луча зрения: $\vec{e}(\sin \gamma \cos \psi; \sin \gamma \sin \psi; -\cos \gamma)$. Тогда $x_E = (z_D - z_E) \tan \gamma \cos \psi$, $y_E = (z_D - z_E) \tan \gamma \sin \psi$.

Далее будет удобно ввести следующие обозначения для разниц высот,

$$h_e \equiv z_E - z_0, \quad h_d \equiv z_D - z_0, \quad h \equiv z_D - z_E = h_d - h_e,$$

и опустить индекс E у X - и Y -координат эльфа,

$$x = h \tan \gamma \cos \psi, \quad y = h \tan \gamma \sin \psi \quad (24)$$

В рамках такой упрощенной модели время пика сигнала в детекторе может быть представлено как $t_0 + SE + ED$, где t_0 – время (начала) разряда, т.е.

$$T_{\text{mod}} = t_0 + \frac{h_e}{\cos \theta} + \frac{h}{\cos \gamma} \quad (25)$$

(для простоты времена и длины приведены к единым единицам измерения, т.е. $c \equiv 1$). Здесь θ – угол, под которым видная точка $E(x, y)$ из источника S . При переходе к полярным координатам, $x_0 = \rho_0 \cos \psi_0$, $y_0 = \rho_0 \sin \psi_0$, и введении направления $\vec{SE}/SE \equiv \vec{e}_s = (-\sin \theta \cos \phi; -\sin \theta \sin \phi; \cos \theta)$, полярные θ и ϕ выражаются через полевые γ и ψ :

$$\tan \theta = \frac{1}{h_e} \sqrt{h^2 \tan^2 \gamma - 2h\rho_0 \tan \gamma \cos(\psi - \psi_0) + \rho_0^2} \quad (26)$$

и

$$\tan \phi = \frac{\rho_0 \sin \psi_0 - h \tan \gamma \sin \psi}{\rho_0 \cos \psi_0 - h \tan \gamma \cos \psi} \quad (27)$$

Сам разряд расположен вдоль луча зрения $\vec{e}_0(\sin \gamma_0 \cos \psi_0; \sin \gamma_0 \sin \psi_0; -\cos \gamma_0)$ с $\tan \gamma_0 = \rho_0/h_d$, и поэтому

$$D\vec{E} = \vec{e} \frac{h}{\cos \gamma}, \quad S\vec{E} = D\vec{E} + S\vec{D} = \vec{e} \frac{h}{\cos \gamma} - \vec{e}_0 \frac{h_d}{\cos \gamma_0} = \vec{e}_s \frac{h_e}{\cos \theta}$$

Отметим также, что

$$x_0 - x = h_e \tan \theta \cos \phi, \quad y_0 - y = h_e \tan \theta \sin \phi \quad (28)$$

При формулировке байесовской модели (см. ниже) будет удобнее ре-параметризовать t_0 , представляя (25) как (вычитая время распространения сигнала от S к D по прямой линии)

$$T_{\text{mod}} = T_0 + \frac{h_e}{\cos \theta} + \frac{h}{\cos \gamma} - \frac{h + h_e}{\cos \gamma_0}. \quad (29)$$

11.3 Величина сигнала в детекторе

Основным недостатком рассмотренной выше модели является отсутствие динамической информации о развитии свечения. Величина сигнала в детекторе чувствительна к ориентации разряда. В дипольном приближении диаграмма направленности ЕМР, сгенерированного разрядом, имеет фактор $\sin^2 \chi$, где χ – угол между направлением электрического диполя¹⁵ и направлением распространения излучения. Дипольное излучение в дальней зоне имеет распределение интенсивности (в единицах СГС)

$$dI = \frac{L^2 |\dot{I}(t)|^2}{4\pi c^3} \sin^2 \chi d\Omega_s,$$

где L – характерная длина молниевых разряда («длина диполя»). Если связать с источником ЕМР сферическую систему координат с углами θ, ϕ , то

$$d\Omega_s = d\phi \sin \theta d\theta = \frac{\cos^3 \theta}{h_e^2} dx dy,$$

причем $\sin^2 \chi = 1 - (\vec{e}_s \cdot \vec{e}_\alpha)^2$, где $\vec{e}_s = (-\sin \theta \cos \phi; -\sin \theta \sin \phi; \cos \theta)$, а \vec{e}_α – (единичный) вектор ориентации диполя. Вводя (по аналогии с θ, ϕ) угловые координаты направления диполя, $\vec{e}_\alpha = (-\sin \alpha \cos \phi_\alpha; -\sin \alpha \sin \phi_\alpha; \cos \alpha)$, получаем

$$\cos \chi \equiv (\vec{e}_s \cdot \vec{e}_\alpha) = \cos \theta \cos \alpha + \sin \theta \sin \alpha \cos(\phi - \phi_\alpha)$$

Таким образом, мощность электромагнитного излучения, доставляемого ЕМР на «горизонтальную площадку» эльфа, может быть представлена в виде

$$dP_{\text{ЕМР}} = \frac{L^2 |\dot{I}(t)|^2}{4\pi c^3 h_e^2} \sin^2 \chi \cos^3 \theta dx dy,$$

¹⁵Точнее, с направлением его второй производной, совпадающей с направлением скорости изменения тока разряда.

Будем считать отклик ионосферы на «подогрев» энергией ЕМР линейным, т.е. что доля ϵ_{ion} (emissivity) от dP_{EMR} изотропно переизлучается. Тогда сигнал в детекторе $dE_d \propto \epsilon_{\text{ion}} \tau dP_{\text{EMR}} \cdot S_d \cos \gamma / DE^2$ (здесь S_d – площадь светосбора детектора, τ – время интегрирования сигнала), $DE = h / \cos \gamma$, а следовательно

$$dE_d = \frac{\tau \epsilon_{\text{ion}} \epsilon_d S_d L^2 |\dot{I}(t)|^2}{4\pi c^3 h_e^2 h^2} \sin^2 \chi \cos^3 \theta \cos^3 \gamma dx dy = \frac{\tau \epsilon_d \epsilon_{\text{ion}} S_d L^2 |\dot{I}(t)|^2}{4\pi c^3 h_e^2} \sin^2 \chi \cos^3 \theta d\Omega_d, \quad (30)$$

где в ϵ_d были включены все (не)эффективности регистрации сигнала (включая эффект PSF и чувствительности каналов), а также был введен элемент телесного угла в поле зрения детектора

$$d\Omega_d = d\psi \sin \gamma d\gamma = \frac{\cos^3 \gamma}{h^2} dx dy.$$

Для получения оценки сигнала в канале достаточно подставить вместо $d\Omega_d$ характерный размер его поля зрения ω_{pix} . Это можно сделать либо зафиксировав его угловой размер, $\omega_{\text{pix}} = \gamma_{\text{pix}}^2$, либо воспользовавшись приближением идеальной оптики и квадратного пикселя со стороной a_{pix} , $\omega_{\text{pix}} = a_{\text{pix}}^2 \cos^3 \gamma / f^2$, где f – эффективное фокусное расстояние. (В предположении малых углов γ обе оценки слабо отличаются друг от друга.)

Таким образом, распределение сигнала по каналам детектора может быть описано как

$$A = A_0 \frac{H^2}{h_e^2} \sin^2 \chi \cos^3 \theta, \quad A_0 \equiv \frac{\epsilon_{\text{ion}} \epsilon_d S_d L^2 |\dot{I}(t)|^2}{4\pi c^3 H^2} \tau \omega_{\text{pix}} \quad (31)$$

где для обезразмеривания сигнала была введена некоторая характерная высота H – в качестве нее может фигурировать, например, высота орбиты спутника, $H = z_D$. В таком виде A_0 включает только несущественные при нашем рассмотрении множители. Точнее, это справедливо в случае фиксированного размера ω_{pix} , в ситуации же идеальной фокусирующей оптики сигнал в детекторе удобнее представить в виде

$$A = \tilde{A}_0 \frac{H^2}{h_e^2} \sin^2 \chi \cos^3 \theta \cos^3 \gamma, \quad \tilde{A}_0 \equiv \frac{\epsilon_{\text{ion}} \epsilon_d S_d L^2 |\dot{I}(t)|^2}{4\pi c^3 H^2 f^2} \tau a_{\text{pix}}^2 \quad (32)$$

В заключение обобщим нашу модель на ситуацию, когда диаграмма направленности излучающего разряда имеет произвольный (но параметризованный!) вид $\eta = \eta(\theta, \phi, t | \Theta_s)$. С помощью этой функции учтем также не-мгновенность разряда, т.е. тем или иным образом параметризуем изменение тока со временем. (В принципе, диаграмма направленности также может меняться со временем.) Например, в ряде работ рассматривается простейшая параметризация $I(t) = I_0 \exp\{-t/(2\tau_s^2)\}$, при которой $\eta \propto t^2 \exp\{-t^2/\tau_s^2\}$ имеет два симметричных максимума (при $t = \pm \tau_s$), а эльф «превращается» в два кольца или в одно широкое кольцо в зависимости от соотношения значений параметров τ_s и τ (времени интегрирования сигнала в канале).

Тогда

$$A(T, \gamma, \psi) = A_1 \frac{H^2}{h_e^2} \eta(\theta, \phi, t) \cos^3 \theta, \quad A_1 \equiv \frac{\epsilon_{\text{ion}} \epsilon_d S_d}{4\pi c^3 H^2} \tau \omega_{\text{pix}} \quad (33)$$

с $t(T, \gamma, \psi) = T - T_0 - h_e / \cos \theta - h / \cos \gamma + (h + h_e) / \cos \gamma_0$. Здесь подразумевается, что θ и ϕ выражены через γ и ψ посредством (26), (27). Для дипольного излучения $\eta = L^2 |\dot{I}(t)|^2 \sin^2 \chi$.

Аналогично, для системы с идеальной фокусировкой

$$A(T, \gamma, \psi) = \tilde{A}_1 \frac{H^2}{h_e^2} \eta(\theta, \phi, t) \cos^3 \theta \cos^3 \gamma, \quad \tilde{A}_1 \equiv \frac{\epsilon_{\text{ion}} \epsilon_d S_d}{4\pi c^3 H^2 f^2} \tau a_{\text{pix}}^2 \quad (34)$$

Ситуация с нетривиальной PSF будет рассмотрена отдельно.

11.3.1 Учет нетривиальной PSF

TODO

11.4 Двумерная модель с наклонным разрядом

Для того, чтобы исследовать чувствительность динамических данных не только к положению, но и к ориентации электрического диполя разряда, рассмотрим сильно упрощенную модель (toy model), в которой будем считать детектор «линейным», т.е. что он наблюдает лишь «плоскую» область пространства атмосферы (вертикальный срез), причем сам диполь разряда лежит в этой же плоскости. Тогда локализацию диполя можно будет охарактеризовать двумя координатам $S(x_0, z_0)$ и углом наклона к вертикали α .

В дальней зоне амплитуда напряженности электромагнитного импульса убывает с расстоянием r как $\propto \sin(\theta - \alpha)/r$, причем $r = h_e / \cos \theta$, где θ – угол направления импульса относительно вертикали. Данная ситуация является частным случаем 3D-геометрии, рассмотренной в предыдущем разделе (с $\psi = \phi = \phi_\alpha = 0$), поэтому в соответствии с (25) и (31) получаем

$$T_{\text{mod}} = T_0 + \frac{h}{\cos \gamma} + \frac{h_e}{\cos \theta} - \frac{h + h_e}{\cos \gamma_0}, \quad A_{\text{mod}} = A_0 \frac{H^2}{h_e^2} \sin^2(\theta - \alpha) \cos^3 \theta$$

причем связь между углами θ и γ упрощается до

$$\tan \theta = \frac{x_0 - h \tan \gamma}{h_e}, \quad \tan \gamma = \frac{x}{h}, \quad \tan \gamma_0 = \frac{x_0}{h} \quad (35)$$

Выражение для T и A иногда будет удобно представить как функции x :

$$T_{\text{mod}}(x) = T_0 + \sqrt{(x_0 - x)^2 + h_e^2} + \sqrt{x^2 + h^2} - \sqrt{x_0^2 + (h + h_e)^2}, \quad (36)$$

$$A_{\text{mod}}(x|\alpha) = A_0 \frac{H^2}{h_e^2} \frac{[\sin \alpha - \cos \alpha \cdot (x_0 - x)/h_e]^2}{[1 + (x_0 - x)^2/h_e^2]^{5/2}} \quad (37)$$

Здесь мы у амплитуды сигнала явным образом указали параметрическую зависимость от α , конечно же, и T , и A также зависят от параметров x_0, h_e (а в случае неизвестной высоты свечения еще и от h).

Локализация области свечения в поле зрения детектора в зависимости от времени¹⁶ представлена на рис. 4 слева в виде графиков $\gamma(T)$ при разных расстояниях до разряда ($x_0 = 0, 50, 100, 150$ и 200 km). Расчеты проведены в предположении $T_0 = 0, z_D = 420$ km, $z_E = 90$ km, $z_0 = 5$ km, т.е. $h = 330$ km, $h_e = 85$ km.

Первое появление сигнала (при $T = T_0$) происходит в точке, лежащей на прямой SD, время распространения света по прямой от источника S до детектора D $T_{SD} = SD = \sqrt{x_0^2 + (h + h_e)^2}$. Также полезно сравнивать T с длительностью одного такта электроники прибора: за такт детектора «УФ атмосфера» $\tau = 2.5 \mu\text{s}$ свет проходит $\Delta T = c\tau = 0.75$ km, а за такт детектора «ТУС» $\tau = 0.8 \mu\text{s}$ – всего 0.24 km.

На рис. 4 справа та же зависимость представлена при разных значения высоты разряда z_0 (горизонтальное расстояние зафиксировано $x_0 = 100$ km). Видно, что временного разрешения приборов вполне достаточно, чтобы различить представленные на рисунке сценарии.

¹⁶Здесь и далее абсолютное время (не отнормированное) выражается в km.

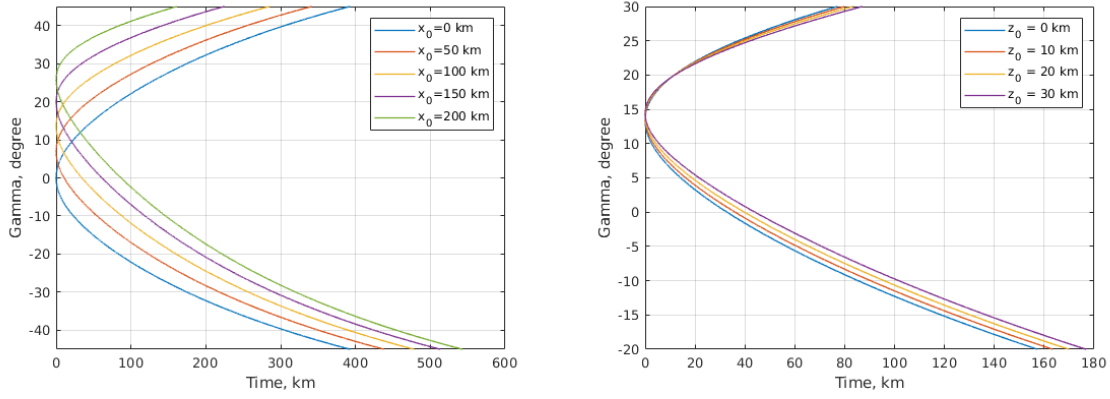


Рис. 4: Зависимость угла наблюдения свечения γ от «детекторного» времени T для различных горизонтальных расстояний x_0 (слева) и различных высот z_0 (справа).

При анализе распределения сигнала в детекторе удобно перейти к безразмерным времени и амплитуде. Нормированную временную координату введем как $t \equiv (T - T_0)/T_{sc}$, а для нормированной амплитуды воспользуемся определением $a \equiv A/A_0 \cdot h_e^2/H^2$.

На рис. 5 представлены зависимости $a(t)$ при четырех ориентациях диполя: вертикальной ($\alpha = 0^\circ$), горизонтальной ($\alpha = 90^\circ$) и наклоненной в ту и другую сторону от вертикали ($\alpha = \pm 45^\circ$). Основная часть свечения заключена в интервале $t \in [0, 0.5]$. Четыре панели соответствуют четырем различным горизонтальным расстояниям от детектора до источника: $x_0 = 0, 50, 100$ и 150 km.

В каждый момент времени «светятся» две точки эльфы – точки пересечения «окружности» фронта ЕМР с бесконечно тонким ионосферным слоем. Лишь в «симметричной» ситуации $x_0 = 0$ (первая панель) при вертикальной и горизонтальной ориентации диполя оба сигнала, $A_1(t)$ и $A_2(t)$, совпадают; во всех остальных случаях присутствует заметный контраст. Если под последним понимать $C = 2(A_1 - A_2)/(A_1 + A_2)$, то для каждого α зависимость $C(t)$ имеет характерный паттерн, позволяющий данным быть чувствительными к ориентации диполя, см. рис. 6.

Другой формат представления сигналов в детекторе изображен на рис. 7. Здесь приведены графики зависимости $A(\gamma)$ из (38), а время t проведено пунктирной линией (вертикальная линия – начало регистрации свечения).

В заключение этого раздела продемонстрируем характер зависимости данных от высоты разряда z_0 . На рис. 8 представлена динамика амплитуды сигнала для наклонного разряда $\alpha = \pm 45^\circ$, регистрируемого при горизонтальном расстоянии $x_0 = 150$ km. В обоих случаях с ростом z_0 временная динамика проходит быстрее, а изображение занимает все меньшую область.

11.4.1 Зависимость от функции тока

На двумерной модели удобно проанализировать и зависимость данных от временной динамики тока разряда. Для простоты рассмотрим лишь один временно профиль тока – гауссовый, $I(t) = I_0 \exp\{-t^2/(2\tau_s^2)\}$. В этом случае выражение для детекторного времени (36) останется без изменения, но T_0 уже будет играть роль не параметра, а переменной (ее значения можно ограничить интервалом $[-3\tau_s, 3\tau_s]$). Для фиксированного положения в поле зрения T_0 является функцией детекторного времени T , поэтому на T_0 можно смотреть как на функцию двух аргументов – положения x точки эльфы в поле зрения и момента T

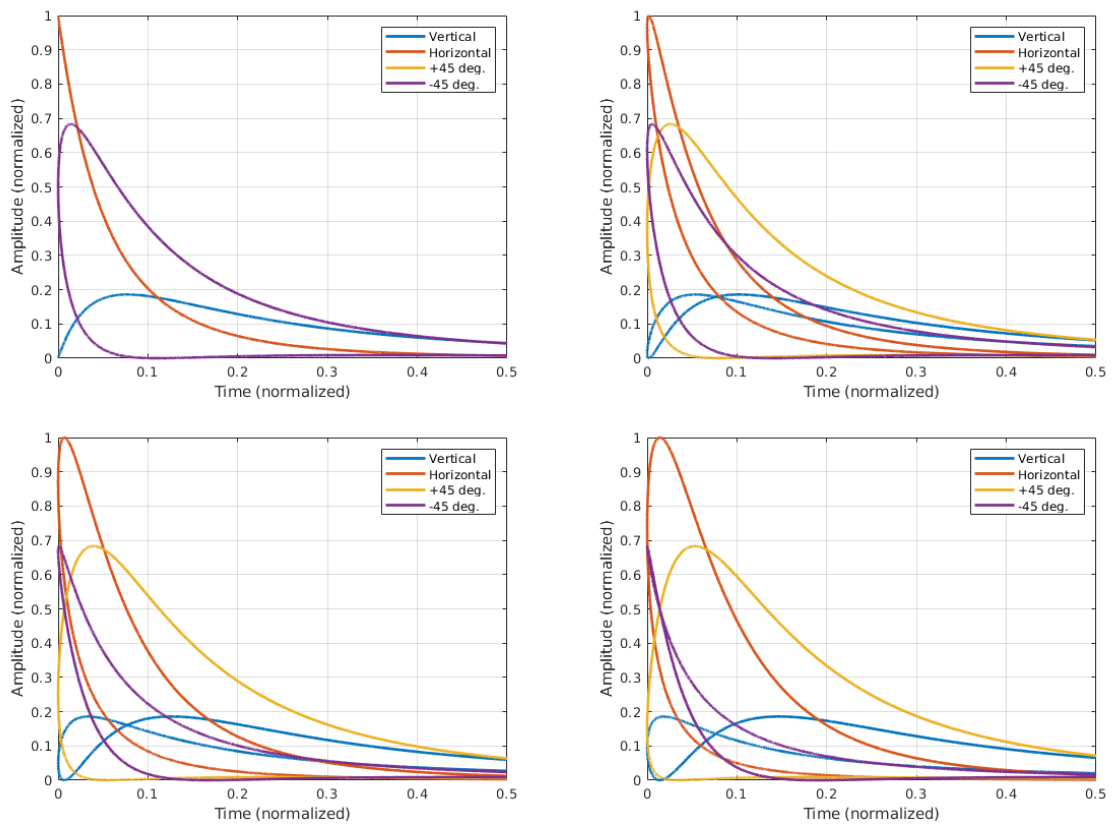


Рис. 5: Зависимость амплитуды сигнала a в детекторе от (нормированного) времени t для четырех ориентаций диполя. Горизонтальное расстояние до разряда (слева-направо, сверху вниз): 0, 50, 100 и 150 km.

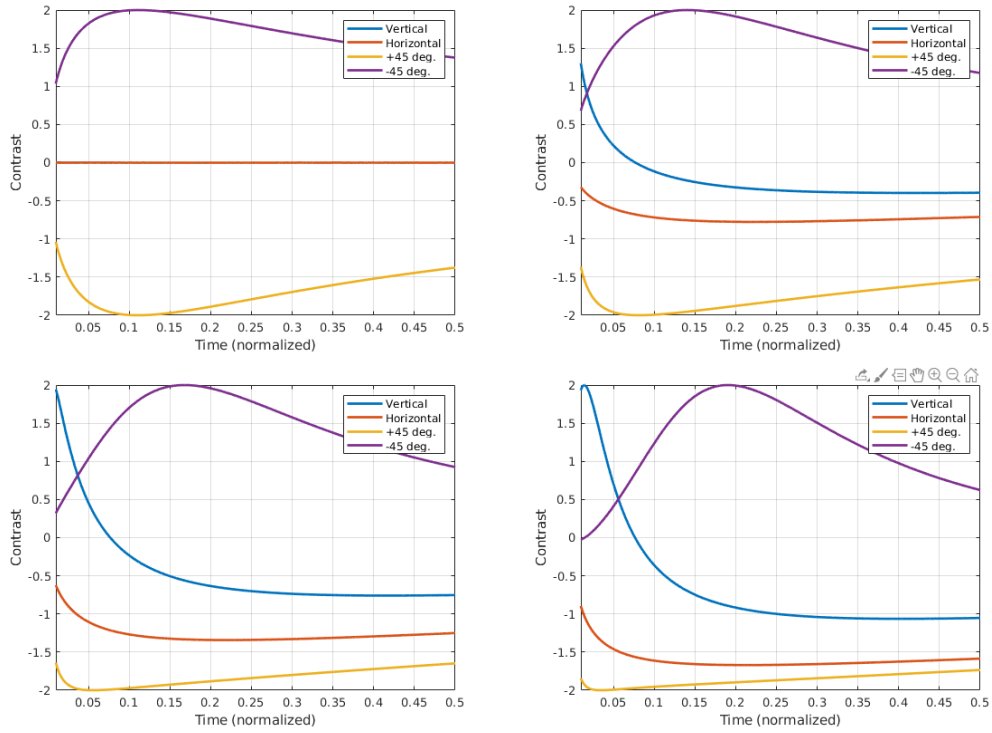


Рис. 6: Зависимость контраста $C(t)$ сигнала, регистрируемого детектором в двух точках эльфа для четырех ориентаций диполя.

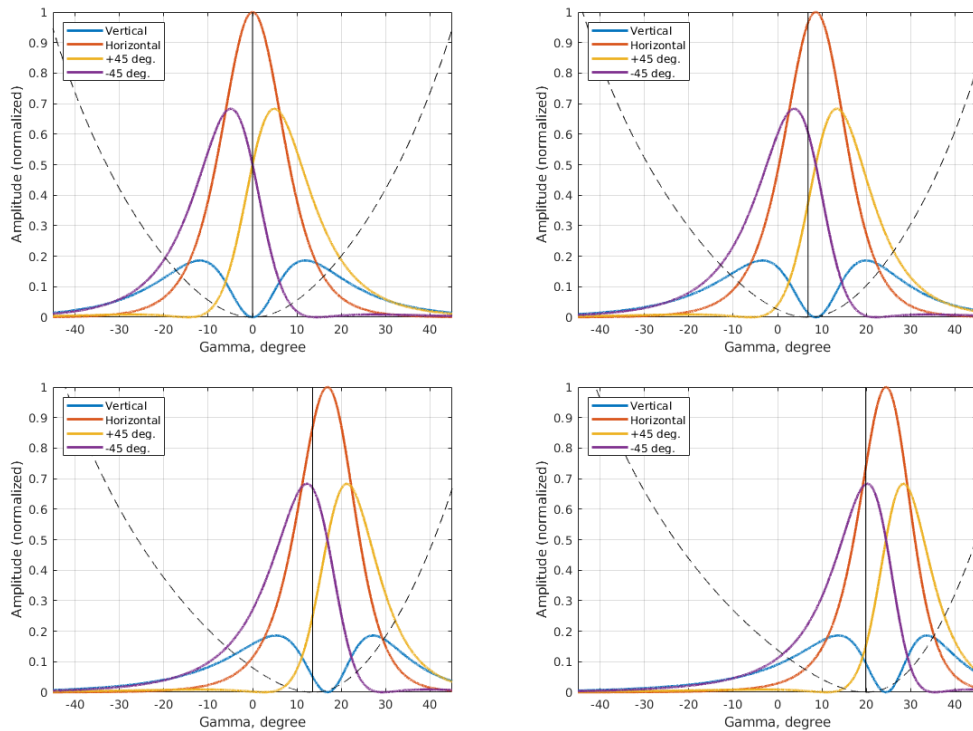


Рис. 7: Зависимость амплитуды сигнала a в детекторе от угла поля зрения γ для четырех ориентаций диполя. Вертикальная линия – момент начала регистрации, пунктиром обозначена зависимость $t(\gamma)$.

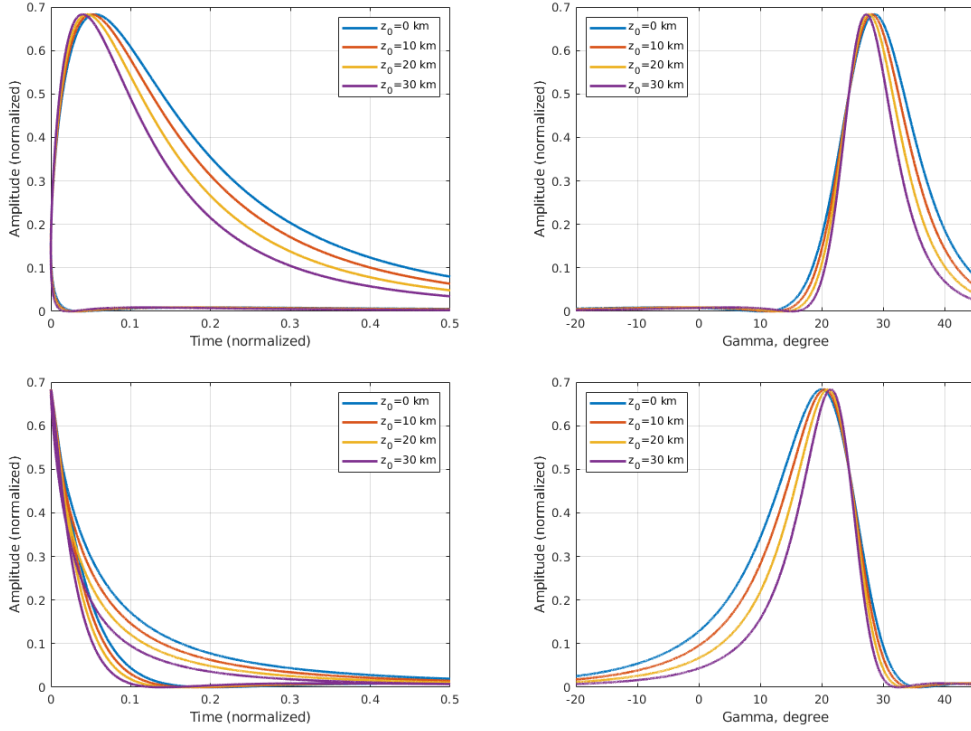


Рис. 8: Зависимость амплитуды сигнала a в зависимости от времени (слева) и полевого угла (справа) при четырех различных высотах разряда, $z_0 = 0, 10, 20, 30$ km, и двух ориентациях диполя, $\alpha = +45^\circ$ (сверху) и -45° (снизу). Горизонтальное смещение фиксировано, $x_0 = 150$ km.

детекторного времени, $T_0 = T_0(x, T)$.

Выражение для амплитуда сигнала должно быть заменено на

$$A_{\text{mod}}(x, T | \tau_s, \alpha) = A_0 \frac{H^2}{h_e^2} \cdot \frac{T_0^2}{\tau_s^2} \exp\{-T_0^2/\tau_s^2\} \frac{[\sin \alpha - \cos \alpha \cdot (x_0 - x)/h_e]^2}{[1 + (x_0 - x)^2/h_e^2]^{5/2}} \quad (38)$$

Для выбранного нами профиля тока легко получить, что максимуму свечения соответствуют два значения $T_0 = \pm \tau_s$. Поэтому в первом приближении можно считать, что мы имеем дублет эльфов конечной длительности $\sim \tau_s$ – две пары разбегающихся светящихся точек, следующих друг за другом с временной задержкой $2\tau_s$.

Интерпретация становится сложнее для малых τ_s , когда разрешение прибора (и угловое, и временное) не позволяет различить эти последовательные пики. Тогда мы имеем ситуацию уширения основного кольца свечения.

Ситуация в чем-то схожа с возникновением дублетов эльфов за счет отражения ЕМР от проводящей поверхности земли. В этом случае также наблюдаются два (концентрических) эльфа с временной задержкой, определяемой разностью временного хода лучей от источника (разряда) и его образа в результате отражения. Но есть и два существенных отличия. Во-первых, источник и его образ расположены на разных высотах в отличие от дублета эльфов, связанного с двумя пиками в (производной) тока разряда. Кроме того, отражение ЕМР не является 100% эффективным, поэтому в этой ситуации второй эльф в дублете должен быть заметно слабее первого¹⁷. Характерная временная задержка в сце-

¹⁷В «токовом» дублете кольца также могут иметь разную интенсивность: для этого достаточно задать разное характерное время нарастания и убывания тока.

нарии с отражением ЕМР в широкой области значений параметров определяется двойной высотой разряда, $\Delta T \sim 2z_0$. Для обычных молний она довольно мала, $\Delta T \sim 0 - 5 \text{ km} = 0 - 30 \mu\text{s}$, и отдельные кольца чаще всего не наблюдаются (см. однако дублиеты «ТУС»). Если же разряд локализован на высоте $\sim 20 \text{ km}$, то временная задержка $T \sim 100 - 120 \mu\text{s}$, что будет зарегистрировано как два четко разнесенных по времени кольца свечения.

Примеры модельных дублетов эльфов по обоим сценариям при разных значениях параметров представлены в следующем разделе.

11.5 Трехмерная динамическая модель эльфа

Рассмотрим теперь динамическую модель эльфа в трехмерном пространстве – ситуацию, которую в полном объеме можно применить как к эльфам «УФ атмосфера», так и к эльфам «ТУС».

11.6 Приложение: генерация данных

На начальном этапе исследования методика проверяется на модельных данных, полученных в результате численного моделирования. Такая модель, получившая название *генерационной*, может иметь разную степень детализации, но в любом случае должна учитывать возможные «флуктуации» сигналов относительно теоретических предсказаний. В нашем случае речь идет о статистическом распределении времен пиков сигналов T_i и их амплитуд A_i в каждом из N каналов изображающего детектора¹⁸.

Наиболее простой (и довольно грубой) проверкой методики реконструкции является выбор генерационной модели тождественной интерпретирующей. В этом случае нам достаточно, задавшись параметрами $x_0, y_0, z_0, \alpha_0, A_0$ (и при фиксированном выборе z_E и z_D), сгенерировать совокупность данных для некоторого набора пикселей. В двумерном случае это означает задать «по-пиксельное» разбиение углов γ_i и получить $\{T_i, A_i\}$ как зашумленные версии уравнений (35), (36), (38).

На следующем уровне моделирования желательно учесть конечность размера пикселя и не-точечность изображения. Оптика детектора может быть охарактеризована функцией рассеяния точки (PSF), например гауссовой с постоянной шириной σ_{psf} . При этом надо будет ввести два индекса: i , идентифицирующий пиксель, и k , нумерующий такт. Соответственно данными выступит набор $\mathcal{D} = \{A_{ik}\}$. Пиксу сигнала в i -м пикселе соответствует амплитуда $A_i = A_{i,k(i)}$ и время $T_i = T_{k(i)}$, где $k(i) = \arg \max_k [A_{ik}]$. «Зашумлять» здесь можно либо итоговые значения A_i, T_i , либо добавить шум непосредственно в величины A_{ik} и после этого предложить тот или иной робастный способ оценки максимума (заменяющий $\arg \max$).

Рассмотрим подробнее двумерный случай. В каждый момент детекторного времени T существуют две точки свечения эльфа $\gamma_1(T)$ и $\gamma_2(T)$, являющиеся корнями квадратного уравнения, получаемого обращением выражения (36). В виду сильного пространственного разнесения этих точек, сигнал от них не может прийти в один и тот же пиксель (в один и тот же момент детекторного времени T). Тогда

$$A_{ik} = \int_{T_k}^{T_k + \tau} dT \int_{\gamma_{\min}^i}^{\gamma_{\max}^i} A(\gamma') \phi_{\text{psf}}(\gamma' - \gamma(T)) d\gamma', \quad (39)$$

¹⁸Точнее, в тех, в которых алгоритм распознавания образа надежно выделяет наличие *активного* сигнала от эльфа.

где из двух должно быть выбрано $\gamma(T)$, наиболее близкое к пикселю $[\gamma_{\min}^i; \gamma_{\max}^i]$. Здесь τ – длительность такта (время интегрирования сигнала), а $\phi_{\text{psf}}(\Delta\gamma)$ – PSF оптической системы, в случае одномерного гаусса $\phi_{\text{psf}}(\Delta\gamma) = \frac{1}{\sigma_{\text{psf}}\sqrt{2\pi}} \exp\{-\Delta\gamma^2/(2\sigma_{\text{psf}}^2)\}$.

Выражения (36), (38) позволяют выразить T и A как функции координаты x , параметрически зависящие от x_0, h, h_e (а A еще и от α). Для этого лучше всего воспользоваться методом Монте-Карло.

На первом этапе выбираются набор изотропно распределенных направлений $\{\theta_n\}$ (по телесному углу, т.е. с равномерными распределениями по $\cos\theta$ и ϕ), причем имеет смысл наложить ограничения, чтобы не генерировать лучи, лежащие далеко от поля зрения. Для этого параллельно рассчитываются соответствующие x_n, y_n и γ_n по формулам

$$x_n = x_0 - h_e \tan\theta_n \cos\phi_n, \quad y_n = y_0 - h_e \tan\theta_n \sin\phi_n, \quad \tan\gamma_n = \sqrt{x_n^2 + y_n^2}/h, \quad \tan\psi_n = y_n/x_n$$

Примечание. Для правильной нормировки в последующем (назначении весов A_n , см. ниже) необходимо оценить величину телесного угла «разрешенной области» Ω_{sim} . К сожалению, аналитически это можно сделать лишь для очень ограниченного набора формы области. Проще всего «вырезать» только ϕ -область, например как $\phi \in [-\phi_{\max}; +\phi_{\max}]$ для некоторого заранее выбранного ϕ_{\max} (его можно попробовать оценить как $\gamma_{\text{pix}}/2$ или $a_{\text{pix}}/(2f)$), и θ -область $\theta \in [\theta_{\min}; \theta_{\max}]$, где $\tan\theta_{\min} = (x_0 - h \tan\gamma_{\max})/h_e$ и $\tan\theta_{\max} = (x_0 + h \tan\gamma_{\max})/h_e$ с γ_{\max} определяемом полем зрения детектора (или чуть шире). В этом случае $\Omega_{\text{sim}} = 2\phi_{\max}(\cos\theta_{\min} - \cos\theta_{\max})$.

Для реализации следующего этапа надо задаться какой-то определенной моделью распределения полей зрения отдельных пикселей детектора. Предположим, что мы выбрали вариант, в котором поле зрения каждого пикселя в угловых координатах представляет собой квадрат со стороной γ_{pix} (т.е. $\omega_{\text{pix}} = \gamma_{\text{pix}}^2$), тогда в плоскости эльфы разбиение по x -бинам может осуществляться как:

$$x^{(i)} = h \tan\gamma^{(i)}, \quad \gamma^{(i)} = i \cdot \gamma_{\text{pix}}$$

Для попадания в линейку должно быть дополнительно выполнено условие $|y_n| \cos\gamma^{(i)} < h\gamma_{\text{pix}}/2$.

Каждому разрешенному лучу n следует приписать вес, равный (вывод этой формулы приведен в следующем разделе)

$$A_n = A_0 \frac{H^2}{h^2} \frac{1}{\tau\omega_{\text{pix}}} \frac{\Omega_{\text{sim}}}{N_{\text{sim}}} \sin^2(\theta_n - \alpha) \cos^3\gamma_n$$

(появление последнего множителя связано с пересчетом от $dxdy$ к $d\Omega_d = \omega_{\text{pix}}$).

Тогда в отсутствии aberrаций достаточно рассчитать двумерную гистограмму A_{ik} с разбиением $x^{(i)}$ и $T^{(k)} = \tau k$, где каждая точка $n(i, k)$ входит с весом A_n : $A_{ik} = \sum_{n=n(i, k)} A_n$.

Примечание. Для построения распределения по бинам рекомендуется воспользоваться функцией `np.histogram2d` (или ее аналогами в других пакетах), позволяющее создавать гистограммы по данным с весами. В качестве весов как раз и будут фигурировать A_n .

Амплитудой сигнала в i -м канале будет та или иная мера максимума A_{ik} по k . При этом надо сигнал предварительно «зашуметь» (например, гауссовым распределением с заданным σ_A или пуассоновскими флуктуациями в случае интерпретации сигнала как счета фотонов), также следует добавить ошибку измерений и в оценку времени T_k (тут вполне подойдет гауссовая ошибка с σ_T).

Генерация сигнала в случае нетривиальной функцией PSF $\phi_{\text{psf}}(\Delta\gamma)$ практически не отличается. Единственное, что надо сделать, это «деформировать» γ_n , полученное на первом этапе: $\gamma_n \rightarrow \gamma_n + \Delta\gamma$, где $p(\Delta\gamma) = \phi_{\text{psf}}(\Delta\gamma)$. При таком подходе можно даже учесть

неэффективность оптики: следует отбрасывать луч с некоторой предопределенной вероятностью $1 - \epsilon_n$, где эффективность $\epsilon_n \equiv \epsilon(\gamma_n)$ в общем случае является (известной) функцией полевого угла.

11.6.1 Трехмерная генерационная модель

На самом деле проще с самого сначала сгенерить данные для трехмерной модели, а уже потом выбрать из нее интересующую нас линейку (или еще какой-то набор каналов фотоприемника, например, принадлежащих одному МАФЭУ). Монте-Карло процедура при этом практически не изменится, с той лишь разницей, что теперь мы будем создавать трехмерную гистограмму A_{ijk} (воспользовавшись функцией `np.histogramdd`), где индексы i, j идентифицируют канал («пиксель»), а k – номер такта.

Кроме того, в данном случае удобнее производить бинаризацию не в плоскости образования эльфы, а непосредственно на фокальной плоскости (ФП) фотоприемника, т.е. пространственные бины совпадут с самими пикселями

$$X^{(i)} = X_{\text{pix}}^{(i)}, Y^{(i)} = Y_{\text{pix}}^{(i)}$$

В простейшем случае одинаковых пикселей (без мертвых зон) разбиение будет эквидистантным, $X^{(i)} = i \cdot a_{\text{pix}} + \text{const}$. При этом, конечно, потребуются реализовать и модель оптической системы, точнее, процедуру пересчета каждого луча зрения \vec{e}_n в точку на ФП. В случае идеальной оптики с фокусным расстоянием f

$$\vec{e}_n(\sin \gamma_n \cos \psi_n; \sin \gamma_n \sin \psi_n; -\cos \gamma_n) \Rightarrow \begin{aligned} X_n &= f \tan \gamma_n \cos \psi_n \\ Y_n &= f \tan \gamma_n \sin \psi_n \end{aligned}$$

Здесь же легко реализуется и гауссова модель PSF

$$X_n \rightarrow X_n + \Delta X, \quad Y_n \rightarrow Y_n + \Delta Y$$

с нормальным распределением (с нулевым средним и с одинаковой или различной шириной) для ΔX и ΔY .

Зададим вес A_n каждого луча так, чтобы суммарное значение весов всех лучей, попадающих в поле зрения одного пикселя ω_{pix} в течение времени интегрирования τ , равнялось бы амплитуде сигнала, определенной ранее в интерпретационной модели (31), т.е.

$$A_n \cdot N_{\text{pix}} = A_0 \frac{H^2}{h_e^2} \sin^2 \chi \cos^3 \theta$$

Здесь $N_{\text{pix}} = \tau N_{\text{sim}} \omega_s / \Omega_{\text{sim}}$, где ω_s – телесный угол из источника, соответствующий телесному углу $\omega_d = \omega_{\text{pix}}$. Так как

$$d\Omega_s = \frac{dx dy}{h_e^2} \cos^3 \theta, \quad d\Omega_d = \frac{dx dy}{h^2} \cos^3 \gamma,$$

то $\omega_s = \omega_{\text{pix}} (h^2 / h_e^2) (\cos^3 \theta / \cos^3 \gamma)$. Поэтому окончательно получаем

$$A_n = A_0 \frac{H^2}{h^2} \frac{1}{\tau \omega_{\text{pix}}} \frac{\Omega_{\text{sim}}}{N_{\text{sim}}} \sin^2 \chi_n \cos^3 \gamma_n \quad (40)$$

Аналогично для сценария идеальной фокусировки (32) с $\omega_{\text{pix}} = (a_{\text{pix}}^2 / f^2) \cos^3 \gamma$

$$A_n = \tilde{A}_0 \frac{H^2}{h^2} \frac{f^2}{\tau a_{\text{pix}}^2} \frac{\Omega_{\text{sim}}}{N_{\text{sim}}} \sin^2 \chi_n \cos^3 \gamma_n \quad (41)$$

В обоих выражениях $\cos \chi_n = \cos \theta_n \cos \alpha + \sin \theta_n \sin \alpha \cos(\phi_n - \phi_\alpha)$. Удобно также выразить веса через физические параметры явления, воспользуемся для этого сразу моделями (33), (34) с обобщенной диаграммой направленности η (для диполя $\eta = L^2 |\dot{I}(t)|^2 \sin^2 \chi$):

$$A_n = \frac{\epsilon_{\text{ion}} \epsilon_d S_d \Omega_{\text{sim}}}{4\pi c^3 h^2 N_{\text{sim}}} \eta_n \cos^3 \gamma_n, \quad \eta_n = \eta(\theta_n, \phi_n, t_n) \quad (42)$$

Напомним, что введение произвольного постоянного фактора H оправдано переходом в интерпретационной модели к безразмерным величинам $A_0, \dot{A}_0, A_1, \dot{A}_1$ и явным выделением в ней зависимости от h_e , веса генерационной модели, выраженные через физические параметры, от H , конечно же, не зависят. В качестве H можно выбрать известную высоту орбиты спутника, $H = z_d$. (Заметим, в A_n не входит множитель $\cos^3 \theta$: он автоматически учитывается при изотропной генерации лучей.)

Приведенные выше формулы (40), (41), (42) легко обобщить на случай, когда детектор ориентирован не в надир, а под некоторым (известным) углом. Также несложно учесть «сферичность» атмосферы.

SSh: Проверить справедливость такой генерационной модели, сравнив с предсказаниями теоретической модели двумерной задачи – см. рисунки выше.

12 Может ли лучшая модель привести к более широкому HDI?

13 Байесовский вывод в SSM

Байесовский вывод возможен не только в низкопараметрических моделях, но даже в ситуации, когда данных, в некотором смысле, меньше чем неизвестных параметров. Однако в этом случае все равно надо как-то сильно ограничить модельную структуру. Но как, если не числом параметров? Ответ прост: раз мы имеем дело с вероятностным заданием параметров (и данных), то надо наложить жесткие ограничения на условные зависимости между ними! Тогда общее совместное распределение $p(\mathcal{D}, \Theta)$ по-прежнему будет относительно простым, несмотря на большую размерность Θ .

Один из способов введения таких вероятностных ограничений представляет собой State-Space Model (SSM) – модель пространства состояний. Структура, заложенная в SSM, довольно проста: есть пространство скрытых состояний, а мы судим о нем путем задания вероятностей переходов из одного состояния в следующее (марковская цепь) и путем проведенного над ним измерения (также заданного вероятностно). Вводя индекс состояния k , имеем

$$p(\mathcal{D}, \Theta) \equiv p(\{y_k\}, \{x_k\}) = p(x_0) \prod_{k=1}^K p(x_k | x_{k-1}) p(y_k | x_k),$$

где $\Theta \equiv \{x_k\}$ – пространство скрытых состояний, $\mathcal{D} \equiv \{y_k\}$ – измеренные данные (те и другие в общем случае многомерные).

В случае простых моделей перехода и измерения SSM позволяет осуществить ВІ аналитически, сводя вычисления к итерационной процедуре (часто ее называют оптимальным байесовским фильтром или аппроксиматором). Так мы приходим к фильтрам и аппроксиматорам Калмана и различным его обобщениям. Однако в более общих ситуациях (нелинейных и не-гауссовых моделях) больше подходят не-оптимальные аппроксиматоры, одним из наиболее универсальных из которых является Particle Filter (PF, в русско-язычной литературе используется термин МЧФ – *мультичастичный фильтр*).

Разберем использование RF на примере задачи реконструкции движения в атмосфере крупного метеороида – болида.

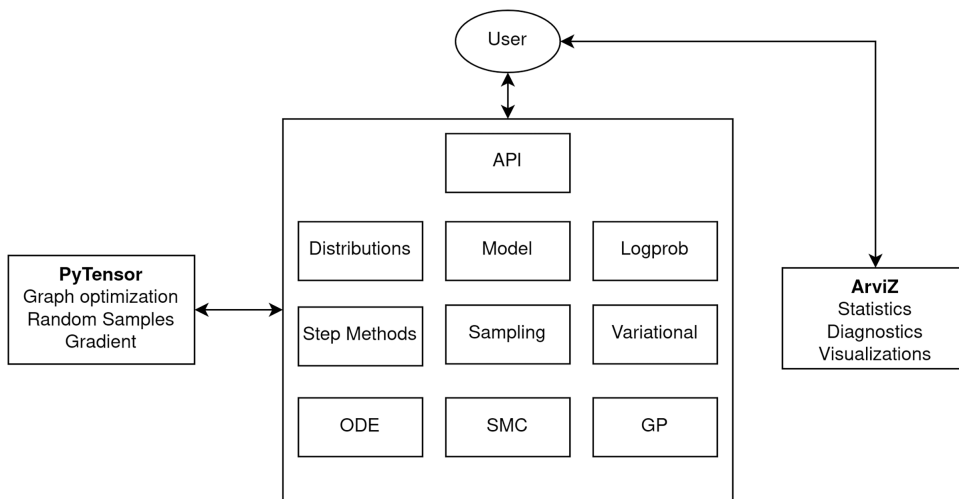


Рис. 9: Архитектура PyMC.

А Введение в PyMC

Кроме многочисленных [примеров](#) и html-версии [книги Освальдо Мартина](#), представленных на сайте разработчиков, в 2023 году вышла обзорная статья, посвященная детальному разбору этого высокоуровневого языка вероятностного программирования:

[\[Abril-Pla2023\]](#) PyMC: a modern, and comprehensive probabilistic programming framework in Python.

В частности, именно из этой статьи взят рис. 9, на котором схематически изображена архитектура основных модулей PyMC. Верхнеуровнево PyMC организован в виде набора (квази)независимых, но взаимодействующих друг с другом модулей. Он содержит такие компоненты, такие как модели и вероятностные распределения и алгоритмы вывода (например, методы MCMC и вариационные приближения), а также продвинутые инструменты моделирования, включая гауссовские процессы (GP) и обыкновенные дифференциальные уравнения (ODE). Архитектура также опирается на две другие библиотеки: PyTensor (вычислительный бэкенд) и ArviZ (для исследовательского анализа байесовских моделей).

Модуль Model предоставляет класс [Model](#), который инкапсулирует все компоненты байесовской модели, задаваемой пользователем. Этот класс служит контейнером для: стохастических случайных величин (random variables), детерминированных преобразований (deterministic variables), наблюдаемых данных (observed data), а также связей между ними, формируя вероятностную графическую модель.

Для удобства пользователей модуль Model включает функции для визуализации направленного ациклического графа (DAG) модели, используя либо GraphViz [\[Gansner2000\]](#), либо NetworkX [\[Hagberg2008\]](#).

А.1 Распределения PyMC

Поскольку байесовская статистика предполагает построение вероятностных моделей, PyMC предоставляет надежные реализации всех распространенных распределений в виде классов в модуле [distributions](#), а также обширный набор более специализированных распределений. Наличие этих классов упрощает создание вероятностных графов для прямого сэмплинга и байесовского вывода.

Стохастические случайные величины в моделях PyMC обладают общим набором свойств,

определенных базовым классом `Distribution`. К ним относятся методы для вычисления: логарифма вероятности (`logp`), логарифма кумулятивной вероятности (`logcdf`), генерации случайных значений.

Распределения условно делятся на непрерывные (`Continuous`) и дискретные (`Discrete`), одномерные и многомерные (`Multivariate`), что наделяет их дополнительными атрибутами (например, формой и типом данных). Помимо этого, алгоритмы вывода PyMC могут оптимизировать работу с распределениями. Например, распределения с ограниченным носителем автоматически преобразуются в вещественное пространство для повышения эффективности сэмплирования.

Для временных рядов (`Timeseries`) доступны сложные многомерные распределения, учитывающие последовательную зависимость между элементами, такие как: авторегрессионные процессы, случайные блуждания (гауссовские и Стьюдента).

Также есть классы-обёртки (`wrappers`), накладывающие ограничения – усечение (`Truncated`) и цензурирование (`Censored`) – на произвольные базовые распределения.

Наконец, если нужное распределение отсутствует в модуле, пользователи могут определить собственное (`CustomDist`).

A.2 Объект `InferenceData` и как с ним работать

На простых примерах мы познакомились с объектом `idata` класса `az.InferenceData`. Его возвращают все сэмплы `pmc`. В частности, в примерах выше были использованы:

```
idata = pm.sample(draws, chains, tune)
idata = pm.sample_prior_predictive(draws, var_names)
idata = pm.sample_posterior_predictive(trace, var_names, predictions=False)
```

Здесь приведены наиболее часто используемые аргументы, если осуществлять вызовы функций вне контекста `with`, то следует еще указать аргумент `model`. Контроль за (псевдо)случайным процессом Монте-Карло осуществляется при помощи аргумента `random_seed`.

В `sample_posterior_predictive` сопоставление ненаблюдаемых переменных модели и постериорных сэмплов `trace` осуществляется на основе имен переменных. Поэтому для постериорного прогностического сэмплирования может использоваться модель, отличная от той, что использовалась в байесовском выводе, при условии, что переменные, по постериорным выборкам которых мы хотим обусловиться, имеют те же самые имена, а также совместимы по форме и размерности.

Эти вызовы заполняют именованные группы объекта `idata`, а именно `prior`, `posterior`, `prior_predictive` и `posterior_predictive`. В обозначениях вероятностной модели с параметрами Θ и данными \mathcal{D} группе `prior` соответствуют сэмплы из распределения $p(\Theta)$, группе `posterior` – из $p(\Theta|\mathcal{D}_{\text{obs}})$, а предсказательные группы представлены сэмплами из $p(\mathcal{D}|\Theta)$ и $p(\mathcal{D}|\mathcal{D}_{\text{obs}}) = \int d\Theta p(\mathcal{D}|\Theta) p(\Theta|\mathcal{D}_{\text{obs}})$.

Примечание. Есть еще три группы, `sample_stats`, `log_lokelihood` и `observed_data`, а иногда добавляется и группа `constant_data`. Группа `sample_stats`, например, нужна при вызове функции `az.plot_pair()`, чтобы указать `divergences`, а вызов `az.compare()` помимо `posterior` требует и группу `log_lokelihood`. Если при вызове `sample_posterior_predictive()` установлен флаг `predictions`, то это подразумевают, что сэмплы получены на новых данных (посредством `pm.MutableData` и `pm.set_data`) и сохранены в группу `predictions`.

Каждая группа является объектом класса `xarray.Dataset`¹⁹ и может быть получена посредством одноименного атрибута, например как `idata.posterior`. Объекты `xarray.Dataset`

¹⁹В `xarray` имплементировано два основных класса-контейнера `DataArray` и `Dataset`. Первый представ-

(также как и `xarray.DataArray`) имеют большой набор полезных методов. В частности, ряд утилит графического отображения являются врезками относительно соответствующей функции `matplotlib`. Например, вызов `idata.posterior.plot.scatter(x=key1, y=key2, color="k", alpha=0.2)` построит диаграмму рассеяния переменных с ключами `key1` и `key2`. Подробнее о функционале `xarray` см., например, [здесь](#).

Чтобы лучше понять широкие возможности, предоставляемые `az.InferenceData`, настоятельно рекомендуем прочитать заметку: [Introduction to xarray, InferenceData, and netCDF for ArviZ](#). В частности, там объяснено, почему в качестве `idata` не используются массивы `numpy` и датафреймы `Pandas`. Некоторые простые способы создания объекта класса `InferenceData` рассмотрены в [Creating Inference Data](#). Большинство групп объекта `InferenceData` имеют predetermined названия первых своих двух измерений (`coordinates` типа `PandaIndex`) – `chain` и `draw`. В операциях с возвращаемым аргументом типа `InferenceData` по умолчанию, если группа не указана явным образом (посредством аргумента `group`), подразумевается `posterior`.

Например, команда

```
idata = az.convert_to_inference_data(<np-1D>)
```

создаст в `idata` группу `posterior` с `chain=1`, а `draw` определится размером одномерного массива, в то время как при

```
idata = az.convert_to_inference_data(<np-nD>)
```

`chain` и `draw` будут заданы в первых двух размерностях (`shape`) `nD`-массива, а последующие размерности сформируют координаты `x_dim_0`, `x_dim_1` и т.д.

Для освоения работы с `InferenceData` крайне полезной окажется заметка [Working With InferenceData](#). Здесь же просто перечислим представленные в ней пункты:

- [Get the dataset corresponding to a single group.](#)
- [Add a new variable.](#)
- [Combine chains and draws.](#)
- [Get a random subset of the samples.](#)
- [Obtain a NumPy array for a given parameter.](#)
- [Get the dimension lengths.](#)
- [Get coordinate values.](#)
- [Get a subset of chains](#)
- [Remove the first n draws \(burn-in\).](#)
- [Compute posterior mean values along draw and chain dimensions.](#)
- [Compute and store posterior pushforward quantities.](#)
- [Advanced subsetting.](#)
- [Add new chains using concat.](#)

ляет именованную многомерную матрицу (`N`-мерное обобщение `pandas.Series`), второй – контейнер объектов `DataArray`, организованный по типу словаря. Подробнее см. [\[Hoyer2017\]](#).

- [Add groups to InferenceData objects.](#)
- [Add Transformations to Multiple Groups.](#)

Полное описание схемы спецификаций этого класса можно найти здесь: <https://python.arviz.org/en/latest/schema/schema.html>.

A.3 Sequential MC в PyMC

Sequential Monte Carlo (SMC) – семейство методов Монте-Карло, широко применяемых в байесовском выводе как для статических, так и для динамических моделей. Среди типичных применений – анализ временных рядов и обработка сигналов (Del Moral, Doucet & Jasra, 2006; Ching & Chen, 2007; Naesseth, Lindsten & Schön, 2019; Chopin & Papaspiliopoulos, 2020).

PyMC поддерживает сэмплирование для статических моделей с использованием SMC с различными ядрами: Metropolis-Hastings, Independent Metropolis-Hastings, Hamiltonian Monte Carlo. Ключевые преимущества SMC в PyMC:

- 1) Работа с мультимодальными апостериорными распределениями (эффективно обрабатывает случаи, когда моды разделены областями с крайне низкой вероятностью).
- 2) Отсутствие необходимости в градиентах (ядро Independent Metropolis-Hastings особенно полезно для моделей без доступного градиента и высокой сложности/размерности, где стандартный Metropolis-Hastings неэффективен).
- 3) Расчёт маргинального правдоподобия (SMC автоматически вычисляет marginal likelihood модели в процессе сэмплирования).
- 4) Поддержка Approximate Bayesian Computation (ABC) – для моделей с неявным правдоподобием, но возможностью симулировать данные по параметрам.

A.4 Гауссовы процессы в PyMC

Одним из наиболее эффективных (и эффектных!) способов получения *моделей в свободной форме* (free form solution на языке D.Sivia) является использование так называемых *гауссовых процессов* (GP). GP детально обсуждаются во всех трех книгах Мерфи: в book0 им посвящена глава 15, в книге book1 – раздел 17.2, а в книге book2 – глава 18.

Специально для этого разработчики PyMC создали модуль `pymc.gp`, подробное описание (API) которого приведено здесь: <https://www.pymc.io/projects/docs/en/stable/api/gp.html>. Кроме того, на сайте часть примеров посвящены освоению GP. Ниже приводится разбор некоторых из них.

При задании GP необходимо выбрать две функции – ковариационную $k(x, x')$ и функцию среднего $m(x)$. Список имплементированных в PyMC ковариационных функций и функций среднего приведен в заметке [Mean and Covariance Functions](#).

A.4.1 GP: базовый вариант

Здесь под *базовым* мы подразумеваем ситуацию, когда и прайор, и правдоподобие являются многомерными гауссовыми распределениями²⁰. Как известно, в таком случае задача вероятностного вывода (а также маргинализация и обуславливание) может быть проведена аналитически. Правда, при этом требуется обращение матриц большого размера (т.е. требует $O(n^3)$ вычислений, где n – число точек данных).

²⁰Конечно же, тут речь идет не о самой функции правдоподобия (как функции параметров модели), а о выборочном распределении.

Эта ситуация подробно разобрана в заметке [Marginal Likelihood Implementation](#).
В базовом варианте речь идет о модели

$$y = f(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \Sigma)$$

причем

$$f(x) \sim \text{GP}(m(x), k(x, x'))$$

Байесовская модель может быть задана в уже маргинализованном по произвольной функции f виде:

$$p(y|x) = \int p(y|f, x) p(f|x) df$$

– эти действия на себя берет метод `pyms.gp.marginal_likelihood` класса `pyms.gp.Marginal`. Только не забудьте задать, посредством наследников классов `pyms.gp.cov` и `pyms.gp.mean`, ковариационную функцию $k(x, x')$ и функцию среднего $m(x)$ (если установленная по умолчанию `pyms.gp.mean.Zero` вам по какой-то причине не подходит). Например²¹,

```
X = np.linspace(0, 1, 10)[: ,None]
with pm.Model() as marginal_gp_model:
    # Specify the covariance function.
    cov_func = pm.gp.cov.ExpQuad(1, ls=0.1)
    # Specify the GP. The default mean function is 'Zero'.
    gp = pm.gp.Marginal(cov_func=cov_func)
    # The scale of the white noise term can be provided,
    sigma = pm.HalfCauchy("sigma", beta=5)
    y_ = gp.marginal_likelihood("y", X=X, y=y, sigma=sigma)
```

Так же элементарно получить и предсказательное распределение значений функции f или ее зашумленной версии – новых данных измерений y . В обоих случаях работает метод `conditional` класса `pyms.gp.Marginal`:

```
# vector of new X points we want to predict the function at
Xnew = np.linspace(0, 2, 100)[: ,None]
with marginal_gp_model:
    f_star = gp.conditional("f_star", Xnew=Xnew)
    # or to predict the GP plus noise
    y_star = gp.conditional("y_star", Xnew=Xnew, pred_sigma=True)
```

Наконец, с помощью метода `predict` можно получить значения условного среднего и ковариационной функции для выбранного сэмпла *SSH: Не совсем понятно – сэмпла чего? Есть ощущение, что речь идет о сэмпле из постериорного распределения параметров*

```
# The mean and full covariance
mu, cov = gp.predict(Xnew, point=trace[-1])
# The mean and variance (diagonal of the covariance)
mu, var = gp.predict(Xnew, point=trace[-1], diag=True)
# With noise included
mu, var = gp.predict(Xnew, point=trace[-1], diag=True, pred_sigma=True)
```

Простой пример применения базового варианта GP приведен в [Example: Regression with white, Gaussian noise](#). Обратите внимание, как производится постериорное предсказательное сэмплирование посредством `pyms.sample_posterior_predictive` и как используется утилита построения специальных графиков `plot_gp_dist` из библиотеки `pyms.gp.util`.

²¹Обращаем внимание, что в PyMC для задания GP массив точек данных должен иметь размерность 2.

A.4.2 GP: негауссово правдоподобие

В базовом варианте, ввиду «гауссовости» и прайора, и правдоподобия, удастся маргинализовать вероятностную модель по (произвольной) функции f . Именно в таком, маргинализованном виде она и была сформулирована выше. Если же по какой-то причине маргинализацию явным образом не проводить (например, при не-гауссовом правдоподобии сделать это аналитически нельзя), то придется работать с другими классами пакета `gp`.

Непосредственная (не маргинализованная) реализация GP в PyMC осуществлена с помощью класса `pymc.gp.Latent` и его методов. Для задания прайора в виде GP придется воспользоваться его методом `prior`. Создание объекта значений функции на новых точках и предсказательного распределения по-прежнему доступно через методы `conditional` и `predict`.

Простой пример использования GP с негауссовым правдоподобием разобран в [Example 1: Regression with Student-T distributed noise](#). Ниже мы приводим байесовскую модель из этого примера:

```
with pm.Model() as model:
    ell = pm.Gamma("ell", alpha=2, beta=1)
    eta = pm.HalfNormal("eta", sigma=5)
    cov = eta**2 * pm.gp.cov.ExpQuad(1, ell)
    gp = pm.gp.Latent(cov_func=cov)
    f = gp.prior("f", X=X)
    sigma = pm.HalfNormal("sigma", sigma=2.0)
    # add 1 because student t is undefined for degrees of freedom less than 1
    nu = 1 + pm.Gamma("nu", alpha=2, beta=0.1)
    y_ = pm.StudentT("y", mu=f, lam=1.0 / sigma, nu=nu, observed=y)
    idata = pm.sample(nuts_sampler="numpyro")

# plot the results
fig = plt.figure(figsize=(10, 4)); ax = fig.gca()
# plot the samples from the gp posterior with samples and shading
f_post = az.extract(idata, var_names="f").transpose("sample", ...)
plot_gp_dist(ax, f_post, X)
```

Чтобы получить значения функции в новых точках, расширим модель²²:

```
X_new = np.linspace(-4, 14, 200)[:, None]
with model:
    # add the GP conditional to the model, given the new X values
    f_pred = gp.conditional("f_pred", X_new, jitter=1e-4)
    # Sample from the GP conditional distribution
    idata.extend(pm.sample_posterior_predictive(idata, var_names=["f_pred"]))

fig = plt.figure(figsize=(10, 4)); ax = fig.gca()
f_pred = az.extract(idata.posterior_predictive, var_names="f_pred").transpose("sample", ...)
plot_gp_dist(ax, f_pred, X_new)
```

Конечно же, PyMC позволяет использовать в качестве параметра правдоподобия не только непосредственно сам GP, но и функции от него. Так в [Example 2: Classification](#) в пара-

²²Аргумент `jitter` метода `conditional` применяется для стабилизации численного расчета.

метре p *бернуллиевского* правдоподобия используется сигмоида:

```
p = pm.Deterministic("p", pm.math.invlogit(f))
```

А.4.3 GP: не-гауссовы процессы и аппроксимации

Разработчики PyMC пошли еще дальше и предусмотрели ситуацию, когда негауссовым может являться сам процесс. Точнее, на данный момент в библиотеке `pmc.py` реализован только *T-процесс Стьюдента*. По сути здесь все происходит также, как и с GP, только вместо `pmc.py.Latent` используется `pmc.py.TP`, который обладает дополнительным аргументом – числом степеней свободы²³. Пример использования TP подробно разобран в [Student-t Process](#).

Кроме того, в пакете `pmc.py` реализовано несколько аппроксимаций гауссовых процессов. Класс `pmc.py.MarginalApprox` аппроксимирует базовый GP (гауссовый процесс с гауссовым правдоподобием) одним из трех способов (указывается в аргументе `approx`): DTC: Deterministic Training Conditional, FITC: Fully independent Training Conditional, VFE: Variational Free Energy.

Еще два класса, `pmc.py.HSGP` и `pmc.py.HSGPPeriodic`, представляют собой так называемое Hilbert Space Gaussian process approximation. Это приближение позволяет понизить количество операций при расчете GP с $O(n^3)$ до $O(nm + m)$, где n – число точек данных, а m – число базисных функций, использованных при аппроксимации. Интересный исследовательский пример применения HSGP можно найти здесь: [Baby Births Modelling with HSGPs](#).

А.4.4 Другие примеры использования GP

Один из наиболее детальных примеров применения GP для анализа и прогнозирования данных рассмотрен в [Heteroskedastic Gaussian Processes](#). Настоятельно рекомендую «продраться» сквозь него.

Оригинальный способ сглаживания измеренных данных с помощью скрытого броуновского движения приведен в примере [GP smoothing](#). Обращаем внимание, что использованное здесь распределение `pmc.py.GaussianRandomWalk` не является GP (хотя пример и разобран в этом разделе): это распределение позволяет моделировать ситуации типа $z_i \sim \mathcal{N}(z_{i-1} + \mu, \sigma^2)$.

²³Еще одно отличие: GP можно складывать друг с другом, а TP нет.

В STAN vs. PyMC

Подробное введение в STAN можно найти здесь: <https://github.com/bob-carpenter/stan-getting-started>.

Байесовский вывод в STAN осуществляется посредством определения нескольких блоков модели, ее компиляции и сэмпирования из постериорного распределения:

- Блок `data` (и `transformed data`): объявление константных данных и данных измерения. В модель данные должны будут передаваться через объект класса `Panda DataFrame`.
- Блок `parameters` (и `transformed parameters`): объявление стохастических переменных (параметров модели).
- Блок `model`: задание распределений стохастических переменных – прайоров и сэмплирующего распределения.
- Блок `generated quantities`: определение переменных, являющихся детерминированными функциями стохастических переменных.
- `model=csp.CmdStanModel(stan_file='file_name.stan')`: компиляция байесовской модели.
- `sample=model.sample(data, seed, iter_sampling, iter_warmup)`: сэмпирование из постериорного распределения.

Для работы с полученными результатами используются функции

- `theta_draws=sample.stan_variable('theta')`: извлечение стохастической переменной и ее «упаковка» в `np.array`.
- `sample.summary(sig_figs = 3)`: получение саммари в виде таблицы.
- `pps_sample=model.generate_quantities(data, previous_fit=sample)`: сэмпирование предсказательных данных (предварительно должна быть сформирована и скомпилирована модель с новыми значениями предикторов и блоком `generated quantities`).

В.1 Полезные примеры STAN

Следующие примеры выбраны из [Case studies](#) и со страниц официального Руководства пользователя [Stan Users Guide](#).

- [The Sum-to-Zero Constraint in Stan](#): использование набора параметров, связанных выражением $\sum_{k=1}^K \beta_k = 0$. (Встроенный `sum-to-zero` вектор позволяет избежать ненужных корреляций, которые появляются, если вводить эту связь посредством $\beta_K = -\sum_{k=1}^{K-1} \beta_k$.) См. также [Built-in sum-to-zero vector](#).
- [Ordered logistic and probit regression](#): использование вектор-параметра типа `ordered` для явного указания упорядоченности входящих в него стохастических переменных (на примере упорядоченной регрессии, использующей набор `cutpoints`).
-

С Как выбирать прайор?!

Рекомендовано к ознакомлению:

- Страничка Aki Vehtari на GitHub: [Prior Choice Recommendations](#)
- Заметка Michael Betancourt: [Towards A Principled Bayesian Workflow](#)
- Статьи:

С.1 Рекомендации по выбору прайора (по Aki Vehtari)

Когда мы говорим, что этот прайор «слабо информативен», мы имеем в виду, что при наличии достаточно большого объёма данных правдоподобие будет преобладать и прайор не будет иметь значения. Однако если данных недостаточно, этот «слабо информативный прайор» будет сильно влиять на постериорный вывод. Фраза «слабо информативный» подразумевает сравнение с плоским прайором по умолчанию. Даже если нет априорной информации непосредственно о масштабе параметров, обычно имеется некоторая информация о масштабе (величины) результатов, которую можно использовать для создания слабоинформативных прайоров.

С.1.1 Независимость

Обычно мы настраиваем наши модели таким образом, чтобы параметры были независимы в своих априорных распределениях. Отчасти это делается для удобства, а отчасти потому, что такая настройка более понятна. Но это означает, что мы должны быть осторожны с параметризацией модели.

Для иерархических моделей можно проверить априорную независимость с помощью постериорной предсказательной проверки (posterior predictive check). Результат такой проверки может побудить нас расширить модель. В этой расширенной модели предположение об априорной независимости также будет разумным, но окажется еще и соответствующим данным. Общее обсуждение вопроса о важности ре-параметризации для байесовского вывода из-за обычного предположения о независимости по умолчанию см. в разделе 5.1 статьи [\[Gelman2004\]](#).

Даже когда мы явно моделируем априорную зависимость, мы обычно используем многомерную модель, такую как прайор LKJ, в которой априорная независимость (диагональная ковариационная матрица) является базовой. Другим примером ре-параметризации является распределение $t(\nu, \mu, \sigma)$. Здесь мы предпочитаем задавать прайор в терминах ν , μ , $\sigma/(\nu - 2)$ или чего-то в этом роде, чтобы учесть тот факт, что масштаб распределения (измеряемый стандартным отклонением или медианным абсолютным отклонением) зависит от ν и σ .

С.1.2 Общие принципы

- Часто, когда люди рекомендуют использовать прайоры по умолчанию, они ограничиваются какой-то версией сопряжённости – нам это не важно!
- Вычислительная цель в Stan: уменьшение нестабильности, которая обычно возникает из-за плохой геометрии постериорной модели, а также из-за тяжёлых хвостов, которые могут привести к плохой адаптации Stan.

- Некоторые принципы, которые нам не нравятся: инвариантность (то есть выбор прайоров на основе принципов инвариантности без понимания смысла параметра), метод Джеффриса, метод максимальной энтропии.
- Слабоинформативный прайор должен содержать достаточно информации для регуляризации: идея состоит в том, что прайор исключает необоснованные значения параметров, но не настолько сильно, чтобы исключать значения, которые могут иметь смысл.
- Слабоинформативный, а не полностью информативный: идея заключается в том, что потеря точности из-за слишком слабого априорного распределения (по сравнению с истинным распределением параметров в генеральной совокупности или текущим уровнем знаний экспертов) менее серьёзна, чем выигрыш в надёжности за счёт включения в модель частей пространства параметров, которые могут быть релевантными.
- При использовании информативных прайоров чётко формулируйте каждый выбор – лучше даже напишите предложение о каждом параметре модели (пример см. в разделе 4.1 статьи [Gelman2016]).
- Как правило, не используйте равномерные прайоры или жёсткие ограничения, если только границы не соответствуют реальным ограничениям (например, параметры масштаба должны быть положительными, а корреляции должны находиться в диапазоне от -1 до 1).
- Используйте сверхслабые прайоры для более точной диагностики основных проблем. Например, прайор $\text{normal}(0,100)$ добавляются буквально для каждого параметра в модели. Предполагается, что всё находится в единичном масштабе, поэтому эти прайоры не будут иметь никакого эффекта – если только модель не выйдет из строя из-за неидентифицируемости. Сверхслабый прайор позволяет увидеть проблемы, не выводя модель из строя.
- При использовании сверхжёстких прайоров следует указывать начальные значения. Рассмотрим следующий сценарий: вы подбираете модель и, чтобы контролировать процесс вывода, устанавливаете для некоторых параметров фиксированные, заданные значения. Теперь вы хотите, чтобы эти параметры были переменными, то есть вы хотите оценить их по данным. Но если вы просто перейдёте к плоским прайорам или даже к слабоинформативным, ваши выводы будут ошибочными, поскольку вам всё ещё нужно понимать некоторые аспекты вашей модели. Итак, вы начинаете с того, что задаёте своим параметрам очень строгие прайоры. (Например, если у вас есть параметр, которому вы изначально задавали значение 4, попробуйте использовать $\text{normal}(4, 0.1)$). Но когда вы это делаете, вам также следует указать начальные значения. Почему? Потому что по умолчанию Stan равномерно распределяет начальные значения. Так что, если вы хотите, чтобы параметр был близок к 4, вам также следует установить начальные значения близкими к 4!

С.1.3 Насколько информативен прайор?

- Прайор часто можно понять только в контексте вероятности. (Или, более обще: в контексте оценочной функции или информации в данных.)

- Априорная предсказательная проверка (prior predictive check) помогает определить, насколько информативна априорная информация о параметрах в отношении масштаба результата.
- После обуславливания данными мы можем проанализировать, насколько чувствительны интересующие нас величины к прайорам и функции правдоподобия (см. например, [Kallioinen2024](#)). Даже очень широкий прайор может оказать влияние, если информация, полученная из данных с помощью функции правдоподобия, слабая.
- Вот идея, которая поможет не запутаться в прайорах по умолчанию: для каждого параметра сравните постериорное стандартное отклонение с априорным. Если постериорное стандартное отклонение для какого-либо параметра более чем в 0.1 раза превышает априорное, сделайте пометку: «Априорное распределение для этого параметра информативно». Затем пользователь может вернуться и проверить, имеет ли смысл прайор по умолчанию для этого конкретного примера.

С.1.4 Масштабирование прайоров по умолчанию в зависимости от стандартной ошибки оценки эффекта

SSH: ПРОДОЛЖЕНИЕ СЛЕДУЕТ...