Selection of $B_s^0 \to \psi(2S)K_{\mathrm{S}}^0$ events

LHCb bei E5

June 24, 2020

Prerequisites

- basic `python` (you will work with `pandas.DataFrames`)
- basic ideas about machine learning
- basic particle physics and statistics

Contents

1 Introduction

Modern particle physics experiments consist of large detectors with elaborate software. The LHCb experiment is one of the four large experiments located at the four interaction points of the LHC at CERN. The LHC collides proton bunches with a centre of mass energy of 13 TeV 40×10^6 times per second during Run 2 (2015-2018). Following the collision, the partons in the protons hadronize into many different particles. LHCb is most interested in hadrons with a charm or bottom quark in order to perform precision measurements of flavour physics processes. These heavy hadrons are unstable and decay so quickly (on human scales) that they do not interact with the detector – for example a b-hadron usually propagates less than a centimetre – but only their lighter long-lived decay products.

Overall, there are plenty of potentially interesting processes happening at a very high rate. Detecting and reconstructing as much relevant information as possible is crucial but challenging. The LHCb experiment uses an elaborate trigger system to select likely interesting events and reduce the great amount of data to a size that is small enough to be stored. The events are only fully reconstructed offline and asynchronously to the data taking. At this point, the amount of data is already very reduced but still overwhelmingly large. Therefore the data used for analysis has usually passed a central selection stage gathering similar decays. And this is where this lab exercise starts.

Often, the events we are interested in (we will now call this signal) are hidden in the data sample such that some further selection is required. We want to find events corresponding to the decay $B_s^0 \to \psi(2S)K_\mathrm{S}^0$ in a genuine LHCb data sample. Sec. 2 gives a quick idea of why this decay is interesting from a physics point of view and how we compute B_s^0 properties without seeing it in the detector. But this is not the main interest of this experiment. Instead, we want to learn how to use a multi-variate classifier to extract the signal candidates from a heavily polluted data sample. A clear aim is defined in Sec. 3. Sec. 4 provides an overview of how this works. Before starting this exercise though, you should also read through the `jupyter-notebook` which serves as a step-by-step guide and provides more detailed information. The important but more complex concepts required in the process are explained in Sec. 5. And finally, Sec. **??** states a few questions that you should research before conducting the experiment in order to discuss them with the your supervisor.

Before you dive into this experiment: the instructions lead to a very basic, un-tuned version of a standard binary classifier and if you are new to machine learning, you will learn a lot by just following these instructions closely. However if you feel curious or ambitious, do not hesitate to deviate from this! There are several different solutions that lead to an equivalent (or frankly much better) result.

2 Physics and reconstruction

The B_s^0 meson (called 'b-s')contains a s-quark and a $\overline{b}$-quark. The $\psi(2S)$ meson (called 'psi-two-s') consists of a $c\overline{c}$ quark pair. And the K_S^0 meson (called 'k-short') is a superposition of $d\overline{s}$ and $s\overline{d}$ states. The decay changes the quark flavour from $\overline{b}$ to $\overline{d}$ and hence must be a weak process in the Standard Model. Fig. 1 shows the leading order Feynman diagram of the decay assuming that the strange quark is not involved but stays a spectator. This decay could for example be used to measure parameters related to CP violation.

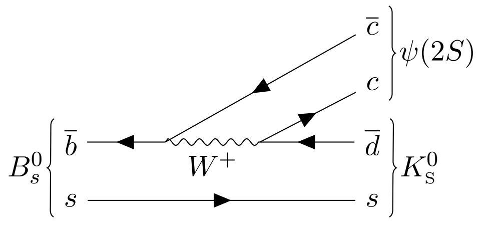

Figure 1: Leading order Feynman diagram of the decay $B_s^0 \to \psi(2S)K_\mathrm{S}^0$ in the Standard Model.

Hint: *you could look at the lifetime or decaytime of the B_s^0 meson in the data sample to convince yourself that this is a weak decay.*

The $\psi(2S)$ meson – like all $c\overline{c}$ states – predominantly decays strongly but has significant electromagnetic branching fractions. Particularly interesting is the decay $\psi(2S) \to \mu^+\mu^-$ because muon tracks are easy to reconstruct and seldom mis-identified. By requiring the invariant mass of the reconstructed $\mu^+\mu^-$ pair to be close to the nominal $\psi(2S)$ mass, the selection excludes other $c\overline{c}$ mesons.
The K_S^0 meson predominantly decays weakly to two charged pions that will be seen and identified by the detector. The superposition of $\overline{d}s$ and $d\overline{s}$ can also be a K_L^0 meson, but the branching fraction of $K_\mathrm{L}^0 \to \pi^+\pi^-$ is more than two orders of magnitude smaller than the branching fraction of $K_\mathrm{S}^0 \to \pi^+\pi^-$ such that we will not worry about those. Also excited neutral kaons, K^* states, can have the quark content $\overline{d}s$. These will not be in our data sample however because all K^* mesons almost exclusively decay to a charged kaon and pion.

In summary, the two final-state mesons are not seen directly by the detector either but only their children. The $\psi(2S)$ meson is reconstructed from two oppositely charged muons with suitable invariant mass, $m(\mu^+\mu^-) \approx m(\psi(2S))$, and the K_S^0 meson is reconstructed from two oppositely charged pions. The data set contains the measured muon and pion variables as well as the reconstructed $\psi(2S)$ and K_S^0 candidates. The variables corresponding to properties of the B_s^0 meson can be obtained from the reconstructed $\psi(2S), K_\mathrm{S}^0$ mesons. The B_s^0 meson properties may also be reconstructed from the $\psi(2S), K_\mathrm{S}^0$ mesons while assuming the true nominal masses of $\psi(2S), K_\mathrm{S}^0$. This assumption is very common and the corresponding variables contain the string `DaughtersConst`.

3 Aim and available samples

You will work with three samples:

- The actual data set after the reconstruction of the LHCb experiment with a very rough selection (see Fig. 2).
- A simulation sample of the signal decay $B_s^0 \to \psi(2S)K_S^0$.
- A simulation sample of the (kinematically) very similar decay $B^0 \to \psi(2S)K_S^0$.

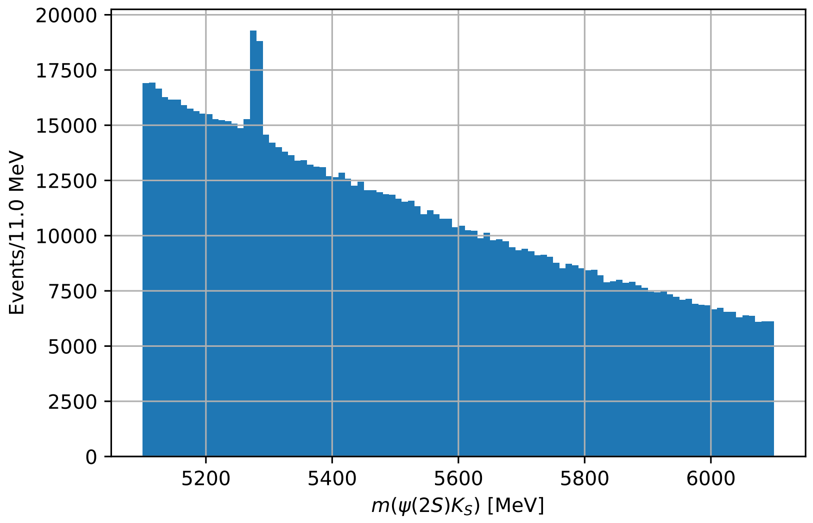

Figure 2: Distribution of the reconstructed invariant mass of the child particles in the full data set.

In particle physics, we are usually interested in one particular decay, like here $B_s^0 \to \psi(2S)K_S^0$. But in this case – like in most cases – the signal is hidden in large amounts of background events (see Fig. 2). The peak does not correspond to our signal but to the much more abundant decay $B^0 \to \psi(2S)K_S^0$. This decay has the same final state and is also kinematically very similar such that pass similar selection requirements and end up in our data sample. The dominating background in this data set is called combinatorial background. For our purposes, we can (and will) assume that there are only three kinds of events in the data sample: the combinatorial background, the visible B^0 peak, and (hopefully) B_s^0 events.

In order to study the physics of $B_s^0 \to \psi(2S)K_S^0$ events, we need to get a cleaner sample with better signal to background ratio. Obtaining such a sample, using a machine learning algorithm to classify signal and background, is the aim of this exercise. We need to define a quantitative measure of the quality of our classifier. In optimization tasks, this is often the minimization of a loss function. Traditionally, we call (the inverse of) this function a figure of merit (FOM) to be maximized. Depending on the priorities – a classification may catch most signal or remove most background –, there are several different FOMs to use. We decide to use the Punzi FOM [1] for the signal significance

$$\mathrm{FOM} = \frac{\epsilon_{\mathrm{sig}}}{5/2 - \sqrt{N_{\mathrm{bkg}}}}\ , \tag{1}$$

with the number of background events in the signal region, N_{bkg}, and the classification efficiency of the signal, ϵ_{sig}. As the signal efficiency can be calculated on signal simulation, this FOM does not rely on the absolute number of signal candidates which is perfect for very polluted data samples like this one.

4 Strategy

This section outlines the signal extraction procedure. It summarizes and complements the `jupyter-notebook` and you should read through both before starting the lab exercise. Some of the steps likely employ methods that are new to you. Sec. 5 explains the most important or advanced concepts in order to focus this section on the process.

4.1 Definition of training samples for signal and background

The dominating background in our data sample is combinatorial. A common choice for a **background training sample** in this case is the upper sideband. The upper sideband of the B_s^0 is the region with reconstructed B_s^0 masses larger than the B_s^0 mass. The sample obtained by simulating the B_s^0 decay helps to identify this region because here we can study what the B_s^0 peak (probably) looks like. The region where we expect most signal is called the signal region/window. This region should never be used in training the classifier to avoid bias.

The simulated B_s^0 events will serve as **signal training samples**. This can become problematic because the simulated distributions sometimes do not match the data distributions. Our simulation algorithms are far from perfect, mainly due to imperfect theoretical physics models and our imperfect modelling of the detector to simulate responses. We could do better, but that would take even more CPU time than it already does. Simulating an event takes several minutes already, and we need millions for every decay we want to study. Some of these mis-modellings can be corrected for by computing weights. The simulation samples you will work with contain a variable called `kinematic_weights` that correct for deviations between data and simulation. They are called kinematic weights because event weighting happens on kinematic distributions, but this is irrelevant to your studies.

Although the weights reduce data to simulation differences in many distributions, some still differ significantly. We will therefore restrict ourselves to features that match well enough after weighting. Since we cannot compare the signal B_s^0 simulation to data yet – because finding the B_s^0 data is the final aim of this project – we use the decay $B^0 \to \psi(2S)K_S^0$ as **control channel**. A control channel is a decay that is reasonably similar to the signal but usually much easier to study. The B^0 decay is (kinematically) very similar to the B_s^0 signal decay and therefore many detector biases or simulation flaws will likely show in a similar manner in both. Additionally, the B^0 decay is very abundant. You can even see its peak in Fig. 2. The pure B^0 data distributions are extracted using an sPlot – the sWeights have already been computed and are in the data sample.

4.2 Feature selection

A classifier can use any number of features to classify input. However, the larger this number, the slower your classifier. With too many features, we may also overfit our training sample. Additionally, not all features provide much information or any additional information at all (ie. knowing the momentum vector, the pseudo-rapidity is redundant information). In order to select good features, we need to choose a metric that evaluates the similarity of distributions. Based on this metric, we can decide what variables in the data sample are suitable because they are not mis-modelled by the simulation (ie. B^0 data and simulation look similar) and find the features that discriminate most between the signal and background training samples.

Hint: *The features that you will select if you follow the instructions in the `jupyter-notebook` are likely not the most meaningful ones. You could construct a set of very meaningful features using techniques called feature extraction that build new features from the existing variables/features* *in a sample. These tools are not very common in particle physics but used much in other fields as they often improve the quality of the classification.*

4.3 Training the classifier and optimization of the classification threshold

The `jupyter-notebook` guides you through the training of k Boosted Decision Trees (BDT) using a k-fold of the training samples. The ROC curve helps evaluating the performance of these BDTs.

Hint: *In regards to the classifier itself and its validation, you are invited to make other choices or use non-basic parameters to optimize your classification or learn something new.*

In order to find the best classification threshold based on the FOM in Eq. (1), we need to compute the signal efficiency and the number of background events in the signal region. The former is easily obtained from the number of signal training samples of the signal region that pass the BDT selection. The latter is a little more tricky because the background training samples are by definition outside of the signal region. The background efficiency ϵ_{bkg} of the classifier should be flat across the invariant mass of the children. Hence, you can estimate it from the background training samples that pass the BDT selection. Assuming the background events strongly outweigh the signal events in the signal region before the BDT selection, the number of background events before the selection is easily estimated from the data sample.

Hint: *There are other and more elaborate ways to estimate the number of signal events in the signal region. Go ahead and try other methods if you like.*

4.4 Finding the signal in the data sample

Eventually, the BDT (or the classifier of your choice) classifies the data sample. This results in a sample with much reduced combinatorial background but a remaining B^0 peak (because the classifier was trained to remove combinatorial background whereas B^0 events strongly resemble signal events) and a visible B_s^0 peak. Last but not least, the number of signal events in the data sample should be determined. To this end, we model the invariant mass distribution of the $K_{\text{S}}^0, \psi(2S)$ combination with two peaks (B_s^0 and B^0 events) and a decreasing exponential function for the remaining combinatorial background. By fitting this model to the data distribution (after the BDT selection), the number of signal candidates can be obtained from the integral of the signal peak model. From here, we can also compute an estimated significance

$$m = \frac{N_{\text{sig}}}{\sqrt{N_{\text{sig}} + N_{\text{bkg}}}}$$

of the observation.

5 Concepts

Extracting the signal from the data set involves a number of advanced statistical and machine learning methods. This section contains quick introductions to the most relevant ones.

Simulation sample/Monte Carlo

Simulation samples (also called Monte Carlo or Monte Carlo samples) are used to study physics or detector effects. Simulation samples are obtained by simulating the proton-proton collisions, the subsequent hadronization, the decay of interest and the resulting detector response. The output format is identical to the detector output of the actual data sample. From here on all reconstruction and selection algorithms are applied in the same way to the simulation sample as they were to the data sample. In the end, the simulation sample represents the expected distributions in the data sample assuming a certain physics model.

Hint: *the simulation samples also contain the true – ie. originally simulated – values of many variables. By comparing them to the reconstructed values you can see how the detector distorts certain distributions.*

Combinatorial background

Combinatorial background consists of events that pass our selection purely due to coincidence. For instance, when two muons (by chance) have a point of intersection, our algorithm will reconstruct them to a $\psi(2S)$ even though the two muons were actually completely unrelated. If the reconstruction algorithm finds an additional random close-by K^0_S, it will combine the K^0_S and the $\psi(2S)$ and we have a reconstructed B candidate. However, these falsely reconstructed B candidates can have almost any invariant mass because the momenta of the random K^0_S and $\psi(2S)$ mesons are uncorrelated. True $B^0_s \to \psi(2S)K^0_\mathrm{S}$ decays on the other hand will have an invariant B^0_s mass close to the nominal mass. The geometry and kinematics of true B^0_s events (ie. momenta, tracks, etc.) have limited degrees of freedom while the same properties of the combinatorial background events are quite arbitrary. Therefore, kinematic variables are especially suited to identify this kind of background.

sPlot

The sPlot technique [2] is a statistical method used to extract one component from a data set with many components. Let's examine a simple example with only two components: signal and combinatorial background. The sPlot requires a variable that clearly discriminates between the two components. In particle physics, this is usually the invariant mass of all child particles in the decay. The signal would here be a peak around the nominal mass of the parent particle. Combinatorial background can often be modelled as a decreasing exponential function. The combined model of the peak and the exponential is fitted to data. The result of this fit presents a proxy of how *signal-like* an event is depending on its invariant mass. The events with invariant mass close to the nominal parent mass are more likely signal than the events far away from it.

An sPlot computes weights according to this fit. These weights are often called sWeights. An event far from the signal peak (unlikely signal) will get a negative weight, whereas an event close to the signal peak (likely signal) will get a (large) positive weight. If done correctly, the sWeighted data distributions will mimic pure signal distributions. The sPlot is an advanced statistical method and there is no need to fully understand all details of this technique for this exercise as the sWeights have already been computed and are contained in the data set. However note that the sWeighted data distribution is only a reliable representation of the true distribution in variables that are uncorrelated with the discriminating variable (here and usually in particle

physics: the invariant mass of the child particles). The sum of the signal sWeights corresponds to the number of signal events in the sample.
You can find a more comprehensive explanation with many pictures here.

Comparing distributions

In this experiment you will need to compare distributions and evaluate their degree of similarity. A good quantitative way of doing this is to compute the distance between the histograms. There are numerous different distances you could use for this. And very often there is not one best way of doing thins. However, you should always carefully investigate whether your problem satisfies all assumptions required for the individual distances to really quantify what you are interested in. A distance you could use for this exercise is the *largest distance between the cumulative probability distributions F^i*:

$$\sup_n |F^1_n - F^2_n| \ ,$$

where n runs over all bins of the histograms. This distance is used in Kolmogorov-Smirnov tests for example.

(Binary) classification

A classifier can be used to identify the belonging of an input to previously defined categories. If there are only two categories, this classifier is also called binary classifier. There are plenty of different classification algorithms suited for very different use-cases. Generally, there are two types of classifiers: supervised and unsupervised learners. In this exercises you will train a supervised learning algorithm. This means that the classifier requires a training sample – in our context: a sample of signal and background events – with known categories. From these samples, the algorithm builds a model of the categories that serves for classification of new unknown input.

k-fold cross-validation

After having trained a classification algorithm, we are usually interested in how ‘good’ the classifier is. That means for example how many background events will end up in the signal category when we classify the actual data sample. Quality criteria computed on the training sample will be biased because these events have already been used to construct the classification model. There are different ways to avoid such bias, one of which is called k-fold cross-validation. Here, the training sample is split into k equally sized sub-samples. The classifier is trained on $k-1$ of these sub-samples and the quality criteria are computed using the kth sub-sample. This process is repeated until each sub-sample was the testing sample once. Using this technique allows to fully exploit the full available training sample and still get unbiased quality criteria (after averaging over the k individual results).

Receiver operating statistic (ROC)

A ROC curve is an illustrative tool to evaluate and compare binary classifiers. Explicitly, the ROC curve represents the true positive rate (tpr) and false positive rate (fpr) for different discriminating thresholds of the classifier output. The diagonal line in the 2D plot of tpr and fpr (from (0,0) to (1,1)) corresponds to a classifier that ‘guesses’ the category of the input. The stronger the curve ‘leans’ towards the side where tpr>fpr, the better the classifier.

6 Colloquium questions

What is a (boosted) decision tree and how does it work?

A decision tree iteratively asks questions to partition data. One decision tree by itself is prone to overtraining. A boosted decision tree is actually a series of simple trees. The first tree is a standard decision tree that classifies some input correctly and other input wrongly. During the boosting, a second tree is added that focuses on healing the wrong classifications of the first. Subsequently, a third, fourth, etc. tree can be added. A BDT with too many trees however is again vulnerable to overfitting.
Gradient boosting is the serial combination of several weak learners (like a simple decision tree) to a powerful one. Bagging on the other hand is the parallel combination of several weak learners (ie. random forest).

How can you tune a BDT?

Learning rate: corresponding to the step size in a minimization. (Too small and you get stuck in a local minimum. Too large and you are may jump over minima.)
Number of estimators: number of 'boostings' (number of sequential trees). If this number is too high, you may overfit the training sample, if this number is too low, you will have a weak classifier (in the extreme case, you only use one decision tree).
Tree depth: number of cuts to reach a leaf. If this number is too high, you may overfit your sample, if this number is too low, you may obtain an undercomplex classifier.

Explain the shape of the distribution of the invariant mass of the children in data

We are looking at reconstructed particles that traversed the detector. We combine them into a B candidate and the mass distribution is the invariant mass of all children in consideration. Combinatorial background makes up most of these events: we wrongly combined things that do not originate from the decay. This means that the invariant mass is quite random. A peaking structure means, that we correctly combined particles that stem from the same parent. The visible peak in our data shows $B^0 \to \psi(2S)K^0_\text{S}$ events. In theory, the invariant mass of the children should be a very sharp peak at the nominal B^0 mass. This peak is smeared out in data due to detector resolution effects.

Why do we want to filter our data? We have all the signal events in the data, any cut will probably remove some!

Our goal in LHCb is to measure properties of the signal, it might be a lifetime, a branching fraction, or a parameter related to CP violation. All of these properties are measured best with a high signal purity as noise (or background) in the data affects/increases the uncertainty of any measurement.

Why should our selection be independent of the invariant mass of the children?

We choose the invariant mass of the candidate as discriminatory variable for several important steps of our selection including but not limited to the signal efficiency (and hence the FOM) and the ratio of signal to background in the sample. These measurements are crucial inputs to the determination of physics parameters, which is beyond this exercise but ultimately the goal. Without a flat efficiency distribution over the mass spectrum, we would not be able to infer the amount of background in the signal area, and we would end up overestimating the signal yield.

Why do we blind the signal region? (That means we ignore the signal region until the very end.) Related to Sec. 4.1.

All events in the signal region are removed before starting any kind of optimization. Otherwise, we might "overtrain" the classifier – be it our brain or a machine – to select as much data in the signal region as possible resulting in a fake peak. This however likely involves abusing statistical fluctuations (hence overtraining) which we must refrain from in all circumstances.

Can a selection give a larger FOM but result in a smaller signal peak? Related to Sec. 4.2.

We mostly avoid overtraining by blinding the signal region and testing the BDT on data not used for its training (cross-validation). Nevertheless statistical fluctuation is not the only pitfall. For the training, we provide the classifier with signal and background labels. These labels however do not only distinguish signal from background but also simulation from data. After all, signal in the training is simulation while background is data. The classifier is thus inclined to train on two kinds of differences: signal-background and simulation-data. As simulation-data differences might be quite big in several variables, we need to make sure that you remove these from the training and only concentrate on features that show no difference between simulation and data.

Why can we not simply use the upper sideband for the number of background events in the FOM? Related to Sec. 4.3.

The background events that drive our uncertainties are the ones we cannot distinguish from the signal, hence the background events in the signal region. We must therefore consider these in the figure of merit. For all practical purposes in this exercise, the background selection efficiency is constant across the invariant mass and we can use that ratio to estimate the background in the signal region even when we are blind. This is not always the case, though!

Why is $N_{\mathbf{sig}}/\sqrt{N_{\mathbf{sig}} + N_{\mathbf{bkg}}}$ a proxy for the significance of the *observation*? Related to Sec. 4.4.

The denominator is similar to the uncertainty in counting experiments (Poissonian uncertainty). The number of signal events divided by the uncertainty of the measurement is hence an indicator for how likely the observed signal peak is just a statistical fluctuation.
An example: we measure $N_{\text{sig}} = 20, N_{\text{bkg}} = 5$ and hence our significance proxy is $20/\sqrt{20+5} = 4$, in particle physics terms: this is better than an *evidence* but not yet a *discovery*. This means that if we repeat the entire experiment and measurement 100 000 times, the resulting number N_{sig} would be within $20 \pm 4 \cdot \sigma = 20 \pm 20$ for 99 994 of these tries or within $20 \pm 3\sigma = 20 \pm 15$ for 99 730 of these tries (approximately, when assuming a Gaussian distribution).

References

[1] Giovanni Punzi. *Sensitivity of searches for new signals and its optimization.* In: eConf C030908 (2003). Ed. by L. Lyons, R.P. Mount, and R. Reitmeyer, MODT002. arXiv: `physics/0308063`.

[2] Muriel Pivk and Francois R. Le Diberder. *SPlot: A Statistical tool to unfold data distributions.* In: Nucl. Instrum. Meth. A 555 (2005), pp. 356–369. DOI: `10.1016/j.nima.2005.08.106`. arXiv: `physics/0402083`.