Developing an automatic pruning utility for statistical models in HistFactory format

Oleksii Kiva under the mentorship of Dr. Alexander Held

Goal: decrease optimization time during max-likelihood estimation of parameters by pruning negligible modifiers from the JSON specification of statistical model, without compromising the relative accuracy of results for pruned and unpruned case.

Statistical model definition

$$f(\boldsymbol{n},\boldsymbol{a}|\boldsymbol{\eta},\boldsymbol{\chi}) = \prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}(n_{cb}|\nu_{cb}(\boldsymbol{\eta},\boldsymbol{\chi})) \prod_{\chi \in \boldsymbol{\chi}} c_\chi(a_\chi|\chi) \quad ,$$

Simultaneous measurement of multiple channels

constraint terms for " auxiliary measurements "

$\boldsymbol{\eta}$ – unconstrained model parameters

$\boldsymbol{\chi}$ – constrained model parameters

$$\nu_{cb}(\boldsymbol{\phi}) = \sum_{s \in \text{samples}} \nu_{scb}(\boldsymbol{\eta},\boldsymbol{\chi}) = \sum_{s \in \text{samples}} \Big(\prod_{\kappa \in \boldsymbol{\kappa}} \kappa_{scb}(\boldsymbol{\eta},\boldsymbol{\chi})\Big)\Big(\nu^0_{scb}(\boldsymbol{\eta},\boldsymbol{\chi}) + \sum_{\Delta \in \boldsymbol{\Delta}} \Delta_{scb}(\boldsymbol{\eta},\boldsymbol{\chi})\Big).$$

multiplicative modifiers

additive modifiers

So far, we have been concerned with the following parameter-constraining modifiers:

Description	Modification	Constraint term c_χ	Input
Normalisation Uncertainty (normsys)	$\kappa_{scb}(\alpha) = g_p(\alpha \mid \kappa_{scb,\alpha=-1}, \kappa_{scb,\alpha=1})$	$\text{Gaussian}(a=0 \mid \alpha, \sigma=1)$	$\kappa_{scb,\alpha=\pm1}$
Correlated Shape (histosys)	$\Delta_{scb}(\alpha) = f_p(\alpha \mid \Delta_{scb,\alpha=-1}, \Delta_{scb,\alpha=1})$	$\text{Gaussian}(a=0 \mid \alpha, \sigma=1)$	$\Delta_{scb,\alpha=\pm1}$

Pruning criterion for normalization uncertainties

$\kappa_{scb,\alpha=0} = 1$

$\kappa_{scb,\alpha=+1}$

$\kappa_{scb,\alpha=-1}$

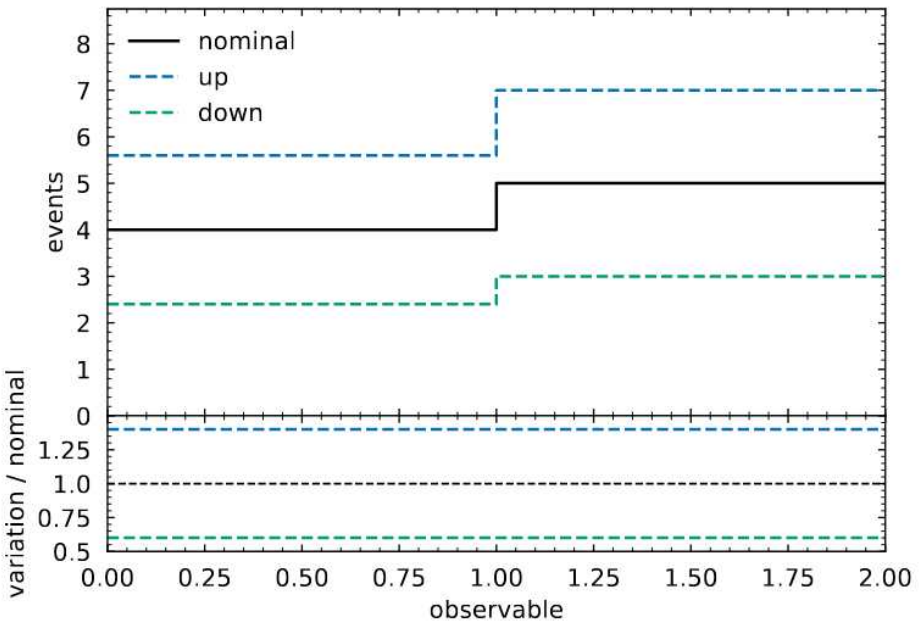
$$max\left(|\,1-\kappa_{scb,\alpha=+1}|,|\,1-\kappa_{scb,\alpha=-1}|\right) \leq \varepsilon$$

An example of a two-bin channel with nominal histogram values ν^{0}_{scb} = [4.0, 5.0]

and up-down variations:

hi = [5.6, 7.0], $\kappa_{scb,\alpha=+1}$ = 1.4

lo = [2.4, 3.0], $\kappa_{scb,\alpha=-1}$ = 0.6



Pruning criterion for correlated shape uncertainties

$\Delta_{scb,\alpha=0} = 0$

$\Delta_{scb,\alpha=+1}$

$\Delta_{scb,\alpha=-1}$

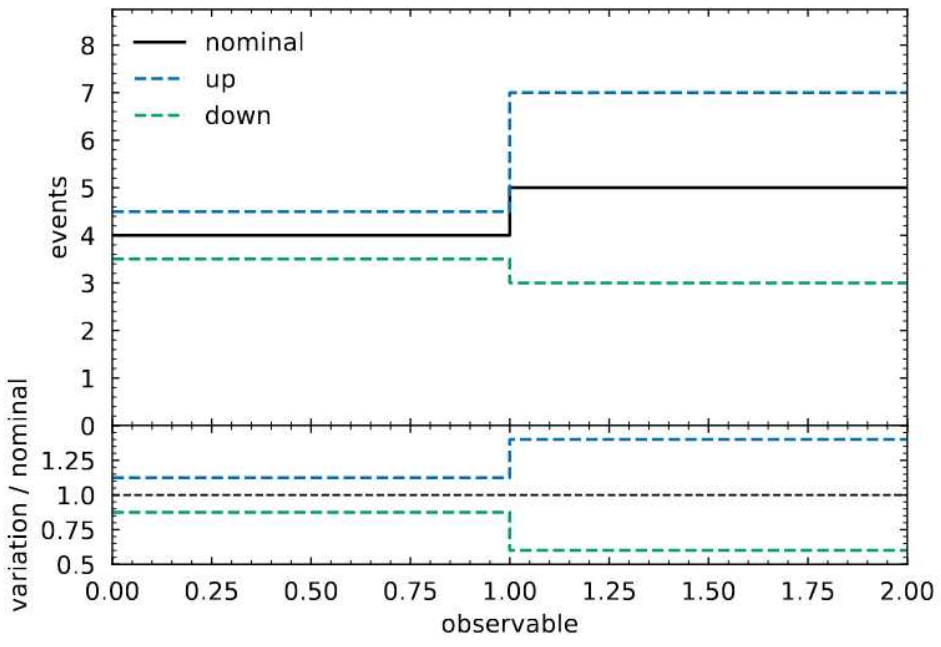
$$\max_{b \in bins} \left(\max \left(\frac{|\Delta_{scb,\alpha=+1}|}{\nu^0_{scb}}, \frac{|\Delta_{scb,\alpha=-1}|}{\nu^0_{scb}} \right) \right) \leq \varepsilon$$

An example of a two-bin channel with nominal histogram values ν^0_{scb} = [4.0, 5.0]

and up-down variations:

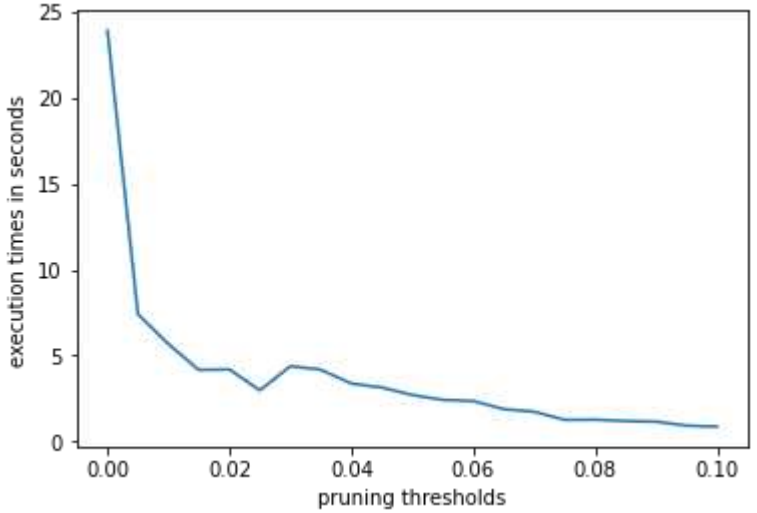
hi = [4.5, 7.0], $\Delta_{scb,\alpha=+1}$ = [0.5, 2.0]

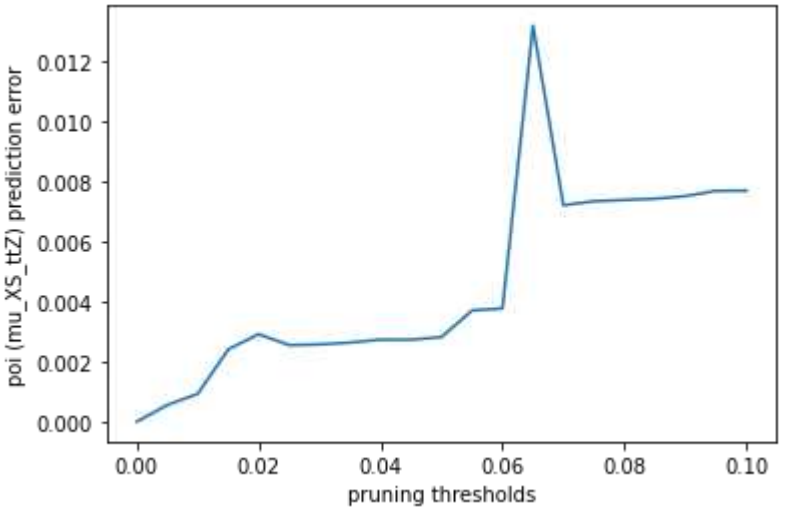
lo = [3.5, 3.0], $\Delta_{scb,\alpha=-1}$ = [-0.5, -2.0]



Measuring an average elapsed time during optimization

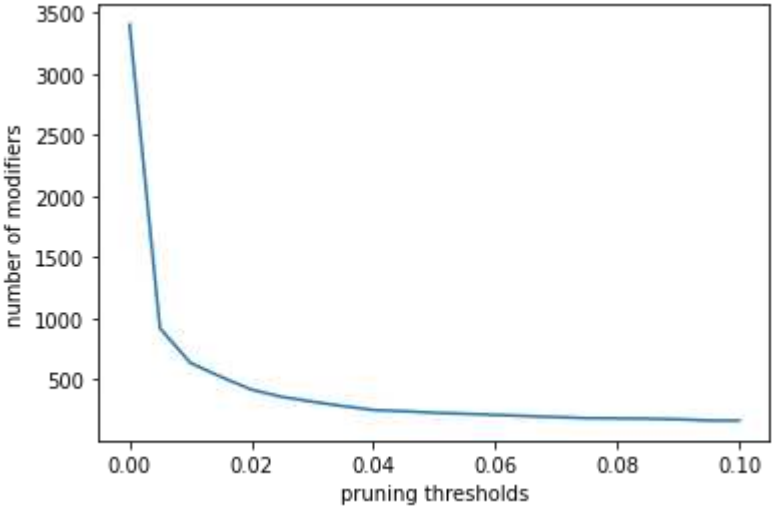
```
average_exec_times = []
exec_times = []
output_params_all_specs = []

for l in range(num_thresholds):

    pruned_data = pruned_workspaces[l].data(pruned_models[l]) #, include_auxdata=False)

    exec_times_pruned = []
    output_params_sigle_spec = []

    print("eps = {} ----------------------------".format(pruning_thresholds[l]))

    for k in range(num_executions):
        t0 = time.time()
        output_params = pyhf.infer.mle.fit(data=pruned_data, pdf=pruned_models[l])
        t1 = time.time()
        exec_times_pruned.append(t1-t0)

        output_params_sigle_spec.append(dict(zip(pruned_models[l].config.par_names(), output_params)))
        print(k+1)
        print(output_params)

    exec_times.append(exec_times_pruned)

    exec_times_pruned = np.array(exec_times_pruned)

    average_exec_times.append(float(np.mean(exec_times_pruned)))

    output_params_all_specs.append(output_params_sigle_spec)
```





Preliminary results of normsys modifier pruning for 4-Lepton workspace

from https://www.hepdata.net/record/resource/2258588?landing_page=True



TO DO:

- try other methods, which (for example) rely on nuisance parameter ranking by their impact on the parameter of interest

- test them, along with described empirical pruning criteria, more extensively on artificial and real statistical model specifications