scalable pythonic fitting

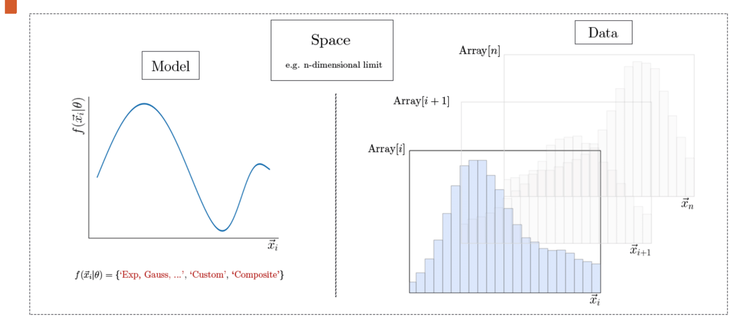
Jonas Eschle (jonas.eschle@cern.ch)
A. Puig, R. S. Coutinho, N. Serra, M. Marinangeli
University of Zurich[UZH]

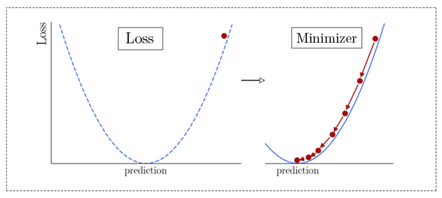
Introduction

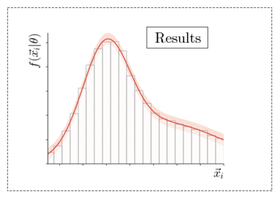
zfit is a likelihood model fitting library in pure Python, well integrated into the scientific ecosystem. It has a focus on customizability and speed and is designed to be powerful enough for analysis in High Energy Physics.

Binned fits are now supported alongside unbinned fits.

Workflow







Unbinned loss

```
obs = zfit.Space("x", limits=(-10, 10))

mu = zfit.Parameter("mu", 1., -4, 6)
sigma = zfit.Parameter("sigma", 1., 0.1, 10)
model_nobin = zfit.pdf.Gauss(mu, sigma, obs)

data_nobin = zfit.Data.from_numpy(obs, normal_np)

loss_nobin = zfit.loss.UnbinnedNLL(model_nobin, data_nobin)
```

Binned loss

```
# make binned
binning = zfit.binned.Regular(50, -10, 10)
obs_bin = zfit.Space("x", binning=binning)

data = data_nobin.to_binned(obs_bin)
model = zfit.pdf.BinnedFromUnbinnedPDF(model_nobin, obs_bin)
loss = zfit.loss.BinnedNLL(model)
```

Minimzation: identical

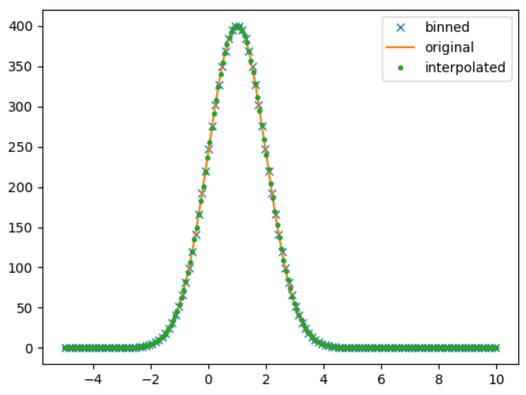
```
minimizer = zfit.minimize.Minuit()
result = minimizer.minimize(loss)  (or loss_nobin)

param_errors = result.hesse()
errors_asym, new_res = result.errors()
```

Binned Model and Mixing

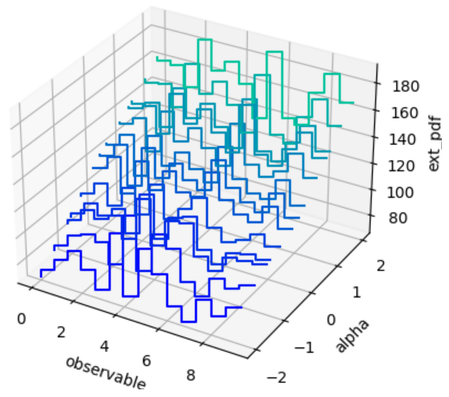
Interpolation of binned

```
spline_gauss = SplinePDF(gauss_binned, obs)
```



Template morphing

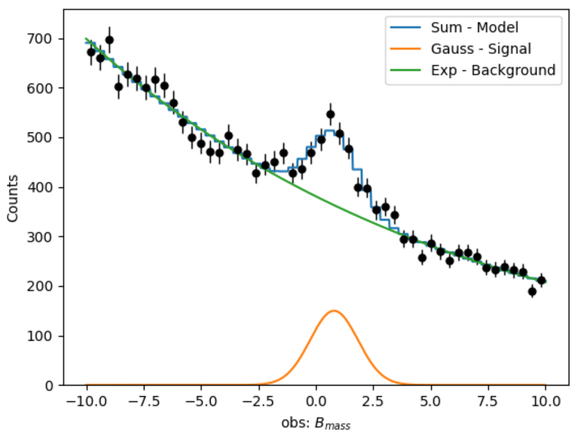
```
pdfs = [...]  # binned template pdfs
alpha = zfit.Parameter('alpha', 0, -1, 1)
morph = SplineMorphing(alpha, hists=pdfs)
```



Composition

Signal and background binned PDFs

```
model = BinnedSumPDF([sigmodel, bkgmodel])
```



Binned data

- To and from hist (boost_histogram)

```
data_binned = BinnedData.from_hist(h)
h = data_binned.to_hist()
```

- Azimov or sampled from PDF

```
azimov_data = model.to_hist()
sampled_data = model.sample()
```

Loss

Simultaneous

```
loss1 = zfit.loss.BinnedNLL(...)
loss2 = zfit.loss.BinnedNLL(...)
loss_sim = loss1 + loss2
```

Constraints

```
constr = zfit.constraint.GaussianConstraint(...)
loss = zfit.loss.BinnedNLL(..., constraints=constr)
```

Minimization

- Wraps minimizer libraries
- Minuit, Scipy, ...

```
minimizer = zfit.minimize.Adam(...)
result = minimizer.minimize(loss)
```

- Convenient BaseClass available

Fit result

- Access results

```
successful = result.converged
mu_result = result.params[mu]
```

- Calculate errors

```
hesse_error = result.hesse()
minos_error = result.error()
```

Performance: TensorFlow

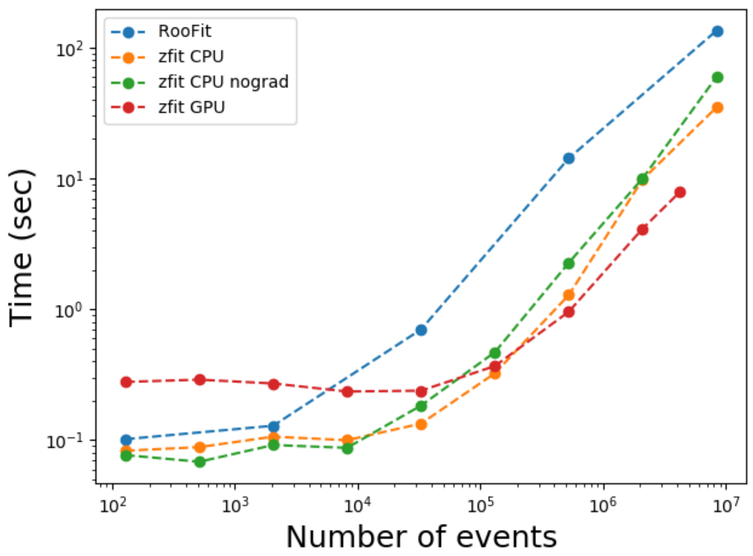
- Python is slow for number crunching
- TensorFlows numpy like API is fast
  - uses kernels for GPU, multi CPU
  - compiles functions just-in-time
  - optimizations, automatic gradient
  - ... yet easy to write like Numpy

```
@tf.function(autograph=False)
def add_mult(a, b, c, d):
    print("compiling...")
    tf.print("running")
    sum_ab = a + b
    sum_cd = c + d
    return sum_ab * sum_cd
```

Typical speedup to Numpy ~30-100

Performance

- sum of 9 Gaussians
- total 2 free parameters



Try it out

Interactive online tutorials are available

https://github.com/zfit/zfit-tutorials

Contributing

Interested to be part of zfit?
Contact us at...
zfit@physik.uzh.ch or on GitHub

Conclusion

zfit provides the possibility of model fitting in pure Python for HEP analyses. Through the Numpy-like TensorFlow interface, very easy to implement performant, customized PDFs. Binned fits allow for large data samples as well as templated fits

Outlook

Custom models and the well defined, decoupled zfit workflow allow to build libraries on top of zfit:

- Higher level fitting libraries
  Amplitude Analysis,...
- Invoke other model building libraries
  Wrapping models, custom Loss,...

[1] The LHCb collaboration, Aaij, R., Abellán Beteta, C. et al. J. High Energ. Phys. (2016) 2016: 104. https://doi.org/10.1007/JHEP02(2016)104