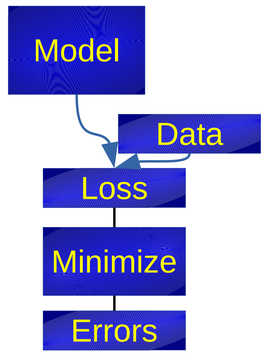
scalable pythonic likelihood fitting

Jonas Eschle (jonas.eschle@cern.ch)
for the zfit team

General likelihood fitting

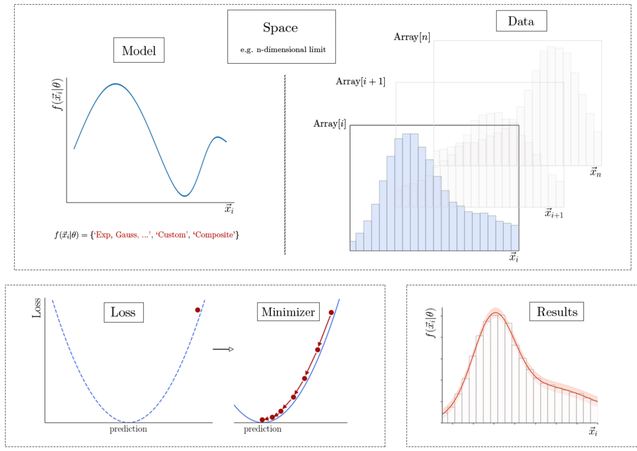
Fitting something more complicated than a normal distribution or a fit is too slow?
zfit allows to build multidimensional, composed models of arbitrary shaped PDFs.
The backend uses speedup-techniques from TensorFlow to speedup and run on GPUs

Used in High Energy Physics with large data & complicated PDFs



Workflow

```
obs = zfit.Space("x", limits=(-2, 3))

mu = zfit.Parameter("mu", 1.2, -4, 6)
sigma = zfit.Parameter("sigma", 1.3, 0.1, 10)
gauss = zfit.pdf.Gauss(mu=mu, sigma=sigma, obs=obs)

data = zfit.Data.from_numpy(obs=obs, array=normal_np)

nll = zfit.loss.UnbinnedNLL(model=gauss, data=data)

minimizer = zfit.minimize.Minuit()
result = minimizer.minimize(nll)

param_errors = result.error()
```



Parameter

Build arbitrary compositions

```
free_param = zfit.Parameter("free", 5, 0, 15)
param_shift = zfit.ComposedParameter("comp",
                  lambda x: x + 5, free_param)
```

Data

- Multiple formats supported
- Full capability of Pandas DataFrames

```
data_raw = zfit.Data.from_root(...)
df = data_raw.to_pandas()
# preprocess in pandas
data = zfit.Data.from_pandas(df)
```

Model building

Custom model

```
class CustomPDF(zfit.pdf.ZPDF):
    _PARAMS = ['alpha']

    def _unnormalized_pdf(self, x):
        data = x.unstack_x()
        alpha = self.params['alpha']
        return tf.exp(alpha * data)

obs = zfit.Space("y", (-4, 4))
custom_pdf = CustomPDF(obs=obs, alpha=0.2)

integral = custom_pdf.integrate(limits=(-1, 2))
sample   = custom_pdf.sample(n=1000)
prob     = custom_pdf.pdf(sample)
```

Composition

```
frac = zfit.Parameter('fraction', 0.5, 0, 1)
sum_pdf = zfit.pdf.SumPDF([gauss, exponential], frac)
```

Product 2-D
(dims defined by observables)

```
#['y', 'x'] <- obs 'y'  *   'x'
product_2d = custom_pdf * sum_pdf
```

Loss

Simultaneous

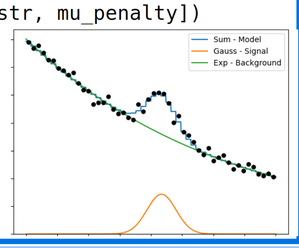
```
nll1 = zfit.loss.UnbinnedNLL(model=gauss1, data=data1)
nll2 = zfit.loss.UnbinnedNLL(model=gauss2, data=data2)
nll_simultaneous = nll1 + nll2
```

Constraints

```
constr = zfit.constraint.GaussianConstraint(...)
mu_penalty = tf.square(mu - 1.3)
nll.add_constraints([constr, mu_penalty])
```

Variety of losses

Binned,unbinned,
chi2, extended



Minimization

- Wraps minimizer libraries
- Minuit, Scipy, Ipyopt, NLOpt

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

- Convenient BaseClass available
- Common convergence criterion
- Automatic gradient available

Fit result

- Access results

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

- Calculate errors

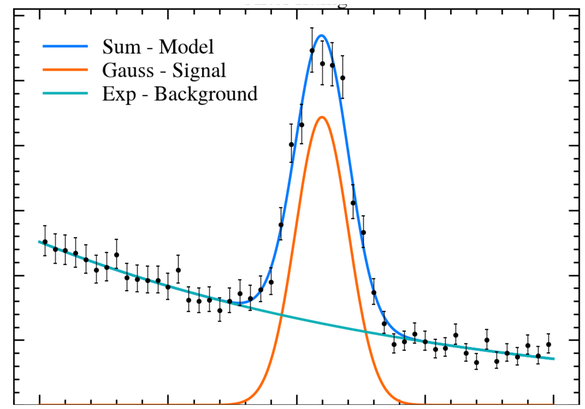
```
hesse_error = result.hesse()
minos_error = result.error()
```

- Store results

```
res_dilled = zfit.dill.dumps(result)
res_loaded = zfit.dill.loads(res_dilled)
```

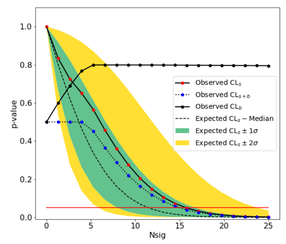
Compositions

Mixture (plot), Convolutions, Products



Inference

hepstats library
Confidence Interval,
limits, sWeights,...



Conclusion

Fast likelihood model fitting in pure Python for complicated distributions.
It allows to connect with many other libraries for a seamless workflow.

SciPy 2024, Tacoma, WA