SVD strip noise simulation

Peter Kvasnicka
Peter.Kvasnicka@mff.cuni.cz

Charles University, Prague

Slides for SVD software meeting
26 September 2018

Overview

Noise in SVD strips

- Noise signals that pass zero-suppression threshold
- Otherwise indistinguishable from signals left by particle

What for...?

- Reproduce noise occupancy in strips
  - This is needed for beam background studies and physics reconstruction
  - We need to know what amount of beam background will seriously hurt
- Improve noise-suppression methods

Noise in SVD strips

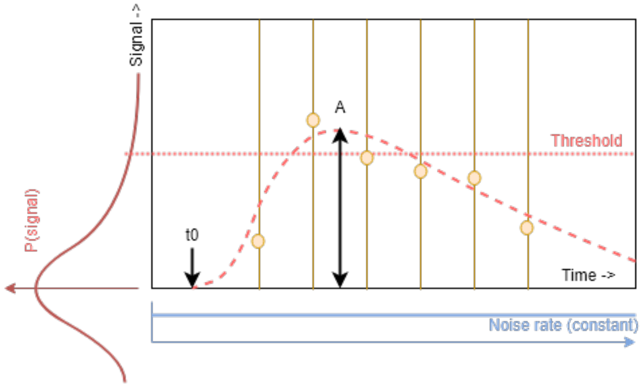


Figure: Noise signal passes zero-suppression threshold when at least one APV25 sample is above threshold.

Noise in SVD strips

Probability of noise signal to pass threshold

The probability depends on:

- Arrival time of the signal
  - We assume Gaussian signals arriving at constant rate.
  - We expect the rate to be sufficiently small so that we don't need to look at autocorrelation structure of the generating noise.
- Signal amplitude
  - We assume noise signals are iid Gaussian with zero mean and standard deviation equal to strip noise.
- Zero-suppression threshold
- Measurement noise
  - We expect noise in the measurement of APV samples. We know it is smaller than strip noise
  - We expect also measurement noise to be Gaussian.
- Waveform shape
  - I only use waveform width as parameter.

Strip occupancy

Strip occupancy is mean number of noise signals seen when the strip is read out:

$$n = \int_{-\infty}^{\infty} dt \int_{0}^{\infty} dA \lambda_0 P(A) P(s_i > T|A, t) \tag{1}$$

where λ_0 is the noise base rate, $P(A)$ is the (half-)Gaussian probability distribution of noise signal amplitude, and $P(s_i > T|A, t)$ is the probability that a signal with size A, arriving at time t, will provide at least one APV25 sample over threshold.
For simulation we need the expression under integral,

$$n(A, t) = \lambda_0 P(A) P(s_i > T|A, t) \tag{2}$$

and, more specifically, we need to be able to draw random samples from it.

Strip noise simulation

We can easily calculate the distribution $P(A)P(s_i > T|A, t)$. Indeed, for each waveform with amplitude A and width w, arriving at time t, we can calculate the probability that one of its samples is above threshold T.
For that, we need to know the measurement noise σ_m, and just calculate the probability of at least one sample over threshold from Gaussian tails of measurement noise distribution (which is 1 minus the probability of all samples being below threshold). We also know the probability of a noise signal with amplitude A from strip noise distribution.

Strip noise simulation

For simulation, we have to generate noise signals with sufficiently large probabilities, so that we don't have to reject candidates too often.
Therefore, we factor out the norm of $P(A)P(s_i > T|A,t)$ - that is, normalize the 2D probability distribution to 1.

$$n(A,t) = \lambda_0 \lambda_{A,t} \Pi(s_i > T|A,t) \tag{3}$$

and use the rate $\lambda_0 \lambda_{A,t}$ to calculate the number of strips on which we generate random noise signals from $\Pi(s_i > T|A,t)$.
Note: The term $\lambda_0 \lambda_{A,t}$ reproduces the $\sim e^{-kT^2}$ dependence of occupancy predicted by H. Spielerś paper and observed in our data. The *Pi* distribution reproduces the signal and time features of strip noise.

Generation of $\Pi(s_i > T|A, t)$ distributions

A-t distributions

The distributions depend on

- Threshold T, 3.0 to 7.0;
- Measurement noise σ_m, 0.1, 0.25, 0.5, 1.0 x strip noise
- Waveform half-width, 100 to 180 ns (200 to 360 ns of β' distribution width)

The distributions are calculated on a grid of A, t values, the distribution is normalized to 1 and norm is stored for each combination of (threshold, sigma, width) parameters.

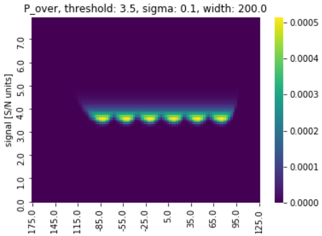
Parameterization of marginal distributions

Marginal distributions are calculated by integrating the 2D distribution along the appropriate axis. They are crucial for simulation. They are parameterized using the following functions:

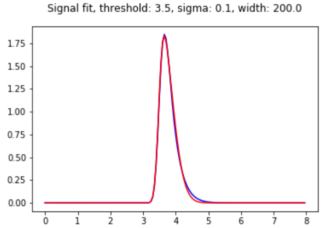
- Signal margin: A single skew-gaussian pdf
- Time margin: A combination of 6 regularly spaced skew gaussians.

Skew-gaussian is a function of the form $2\Phi(\alpha z)\phi(z)$, with Φ and ϕ being the standard Gaussian distribution function and density, and α is the skewness parameter.

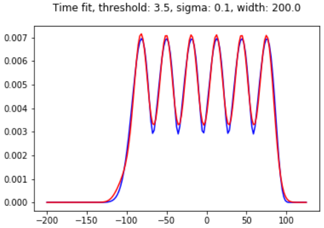
(a) A-t distribution



(a) A margin



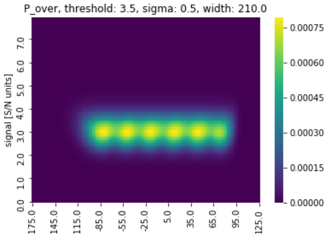
(b) t margin



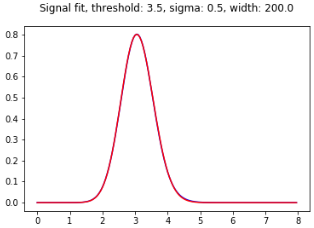
Parameters:

- T = 3.5
- σ_m/σ_0 = 0.1
- w = 200 ns (β' width)

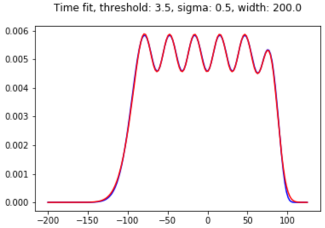
(a) A-t distribution



(a) A margin



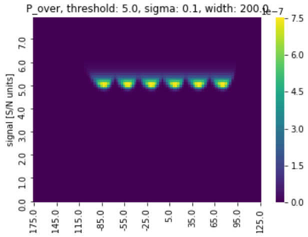
(b) t margin



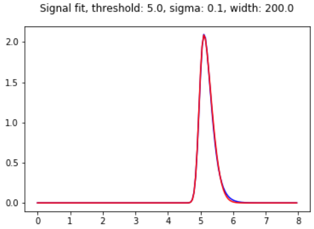
Parameters:

- T = 3.5
- σ_m/σ_0 = 0.5
- w = 200 ns (β' width)

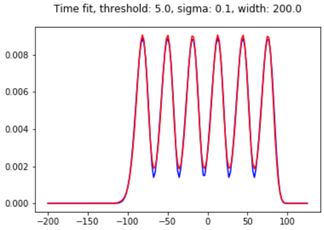
(a) A-t distribution



(a) A margin



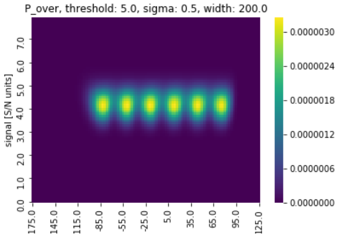
(b) t margin



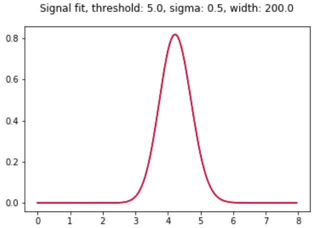
Parameters:

- T = 5.0
- σ_m/σ_0 = 0.1
- w = 200 ns (β' width)

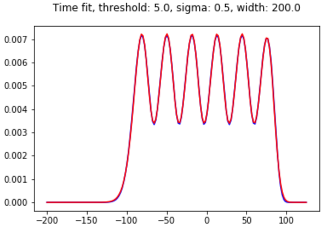
(a) A-t distribution



(a) A margin



(b) t margin



Parameters:

- T = 5.0
- σ_m / σ_0 = 0.5
- w = 200 ns (β' width)

Sampling A-t distributions

General

To draw samples from a bivariate distribution, we need conditional distributions:

$$P(t, A) = P(t)P(A|t) \tag{4}$$

and the same holds for distribution functions, so we take 2 uniform samples u, v and calculate t, A using inverse distribution functions as follows:

$$\begin{aligned} t &= F^{-1}(u) \qquad (5) \\ A &= F^{-1}(v|u) \qquad (6) \end{aligned}$$

If there is a correlation structure between A and t, we have to remember a one-parametric set of distribution functions $F^{-1}(v|u)$ - either parameterize it or use a machine learner.

Sampling A-t distributions

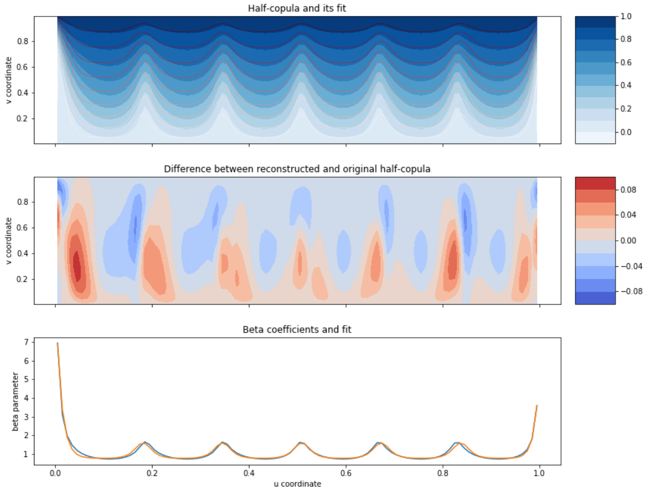
Correlation structure

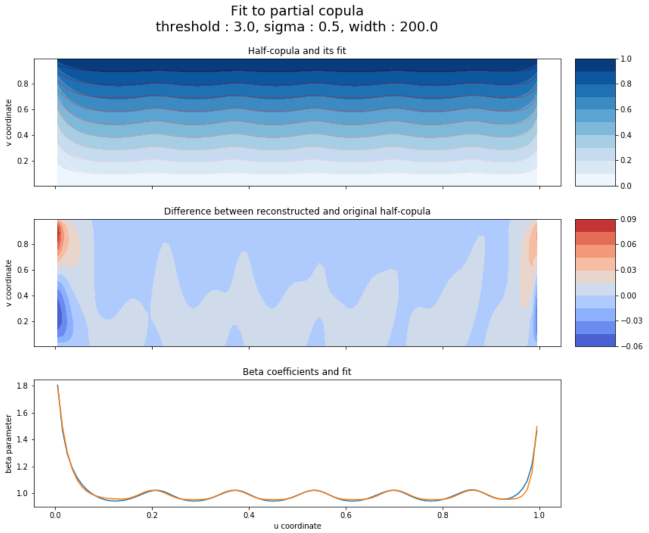
We can view the correlation structure by eliminating everything else - in this case, by eliminating margin distributions. By transforming (t, A) to (u, v),

$$u = F_{t-margin}(t) \quad (7)$$
$$v = F_{A-margin}(A) \quad (8)$$

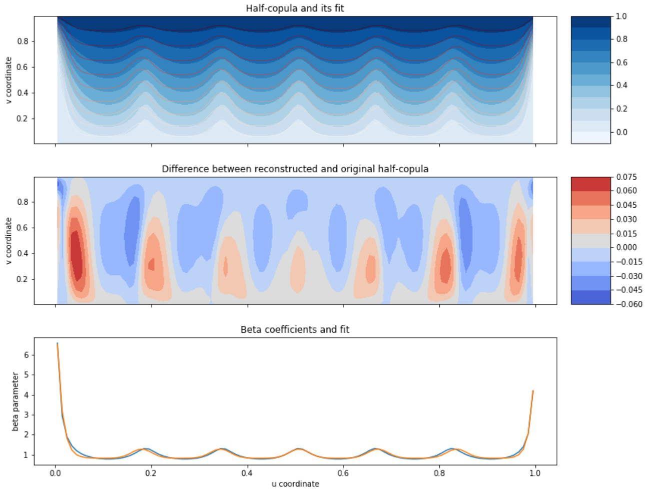
we can construct the set of distribution functions to adjust v for each u to respect the correlation structure - the *partial copula*

Fit to partial copula
threshold : 3.0, sigma : 0.1, width : 200.0

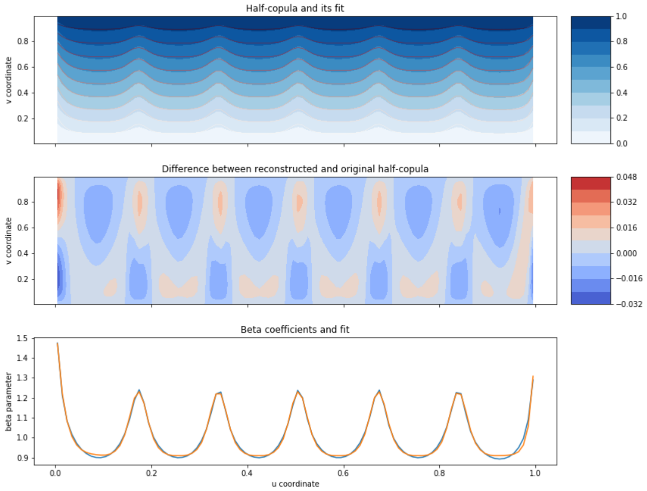



Fit to partial copula
threshold : 3.0, sigma : 0.5, width : 200.0
Half-copula and its fit
v coordinate
0.2
0.4
0.6
0.8
0.0
1.0
Difference between reconstructed and original half-copula
0.09
0.06
0.03
0.00
−0.03
−0.06
Beta coefficients and fit
beta parameter
1.0
1.2
1.4
1.6
1.8
u coordinate
0.0
0.2
0.4
0.6
0.8
1.0

Fit to partial copula
threshold : 5.0, sigma : 0.1, width : 350.0



Fit to partial copula
threshold : 5.0, sigma : 0.5, width : 200.0



Sampling A-t distributions

Implementation: Neural network learner

To streamline the random generation, I use a *neural network random generator*: a neural network that maps the n-tuple (u, v, threshold, sigma, width) to (t, A) pair.

- The training mapping is generated from the half-copula (Python, since a lot of integration and splines are involved)
- The training, handling and use are all delegated to the basf2 mva package framework.
- Norms (noise rates) are fitted to a linear model.

Illustration: Random generator at work

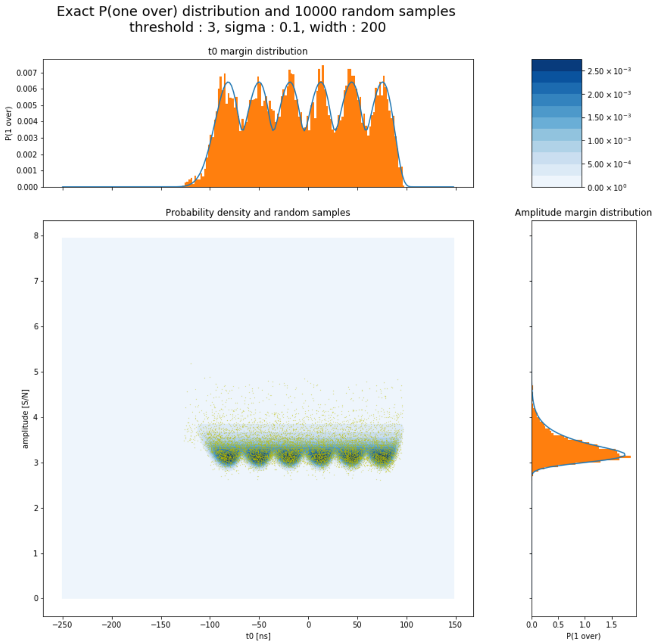


Illustration: Random generator at work

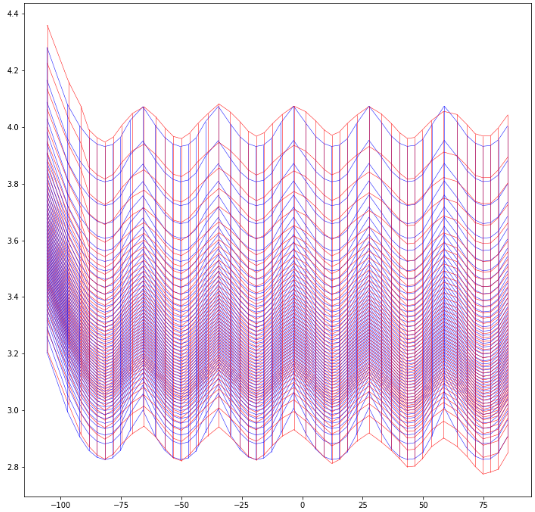


Illustration: Random generator at work

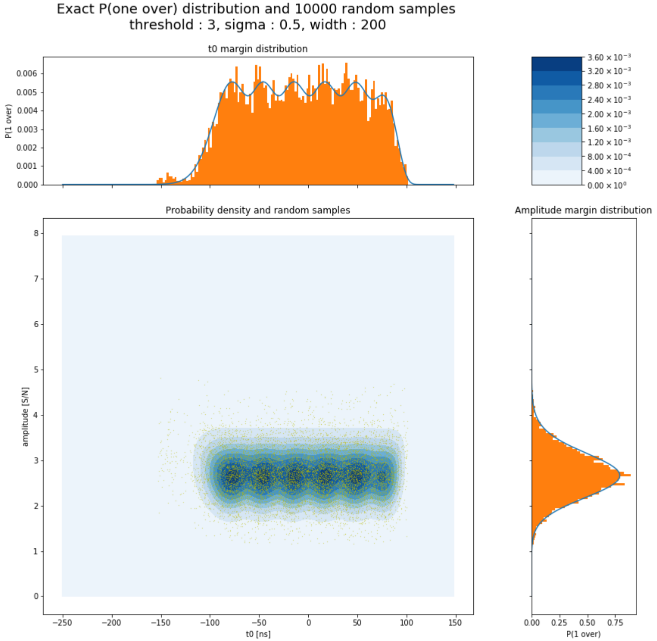


Illustration: Random generator at work

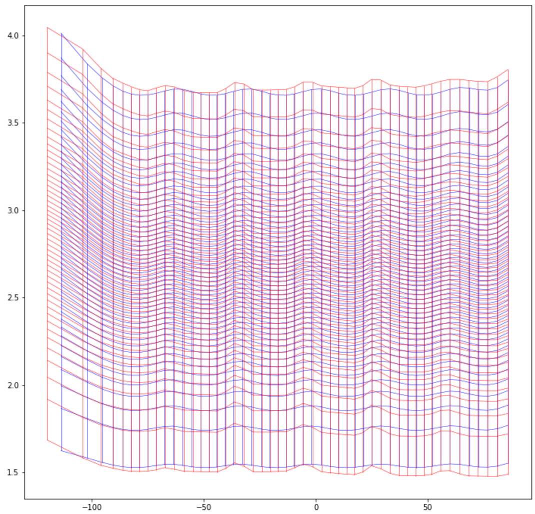


Illustration: Random generator at work

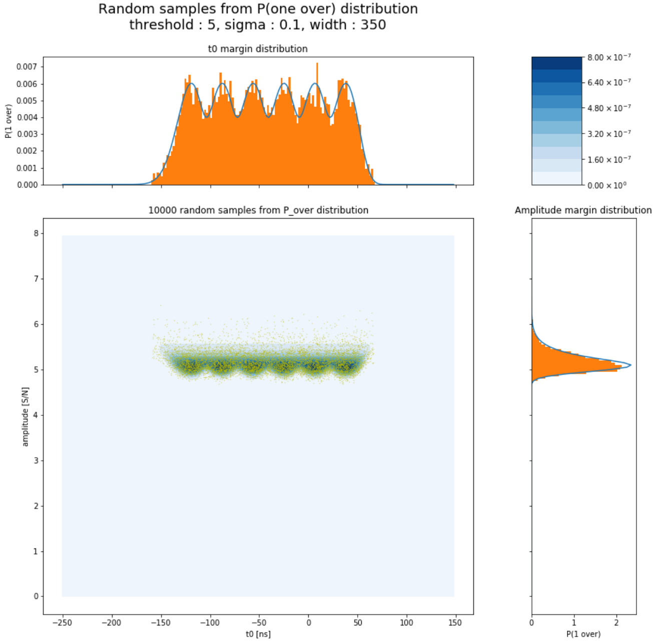


Illustration: Random generator at work

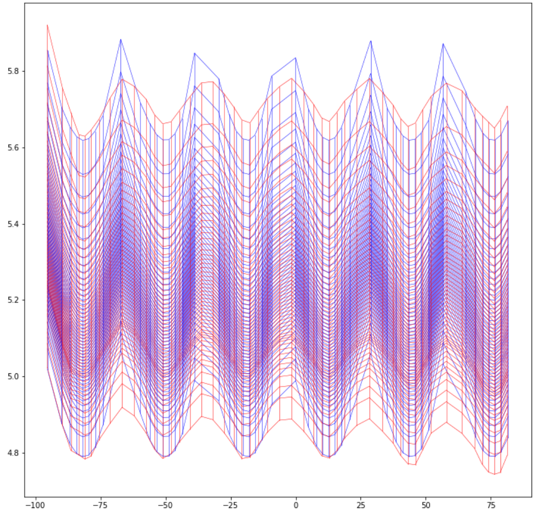


Illustration: Random generator at work

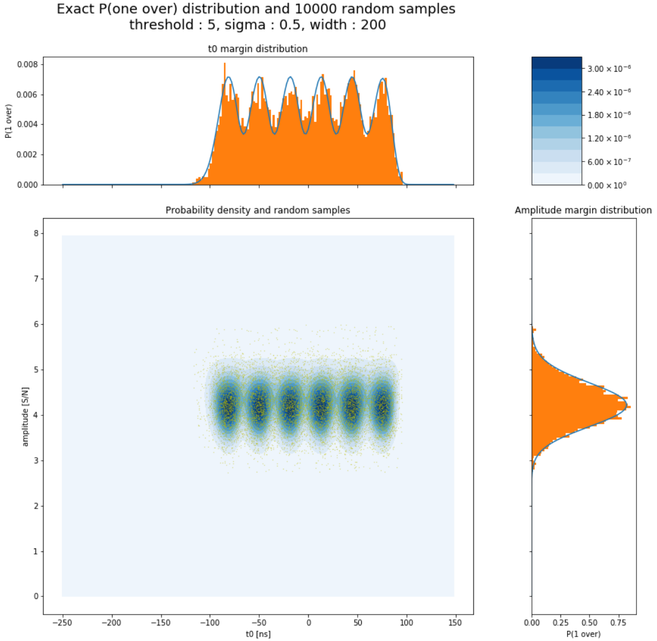
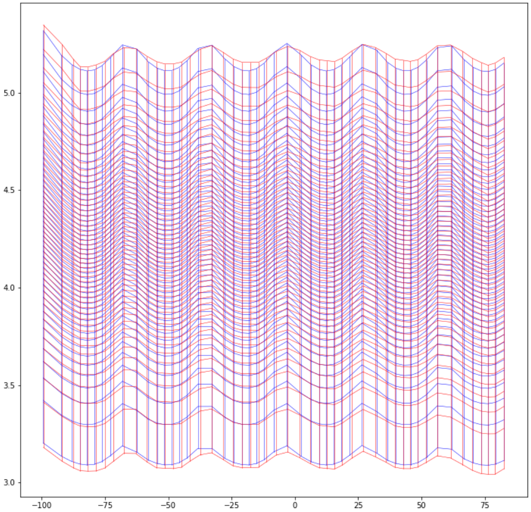


Illustration: Random generator at work



Implementation details

General

To generate random (A,t) pairs with correct probabilities, we need

- Rates: Used to calculate the number of strips on which we actually generate a noise signal. It is base rate times A-t PDF norm, $\lambda_0\lambda_{A,t}$.
  - λ_0 is a strip parameter.
  - $\lambda_{A,t}$ depends on (threshold, σ_m/σ_0, waveform width) combination. Of these, threshold is external and the other two parameters are strip properties.
- (t,A) pairs are generated using neural network, using the same parameters.

. Neural network coefficients are automatically stored in database by mva package.

Current status

- A Python package calculates training data.
- mva package handles network training and management of coefficients.
  - The calculation is from "first principles" and needn't be repeated unless it has to be changed, and the same holds for database content
  - This does NOT hold for base noise rates λ_0, which are true calibration parameters and may need to be updated.
  - Norms are currently calculated using a simple linear fit, and the few coefficients need separate storage in the database. I'm trying to adapt mva to handle this, too.
- I have a small study underway to get proper base rate estimates and make a code that would re-calculate base strip noise rates from data.