Continuous QmisID

Jeremy J.H. Wilkinson

October 2025

1 Introduction

We would like to estimate the probability with which we misidentify the charge of an electron in data. To do so, we use a trick that assumes we can find a region of phase space which is empty unless a charge flip occurs. For example, di-electron truth events near the Z-peak are overwhelmingly produced as opposite charge (OC) pairs due to the Z-resonance, and there are no processes that produce a significant number of same charge (SC) events in truth. Therefore one can be confident that the vast majority of reco events in the SC region originated in the OC region but contained a charge-flip in the reconstruction of one of the truth electrons.

1.1 General strategy

Let us denote the rate at which an OC di-electron event is produced in truth wherein both muons are reconstructed as $\lambda^T(e_+, e_-)$ where $e_{+/-}$ denotes a subset of the properties of the positron/electron. Note that λ^T does not care what charge the electrons are reconstructed as, just that they have in fact been reconstructed. Our assumptions let us write the rate at which we observe reco di-electron events in the OC/SC regions, $\lambda^R_{OC/SC}(e_1, e_2)$, in terms of $\lambda^T(e_+, e_-)$ and the charge flip rate for each electron $\epsilon(e_{+/-})$. The exact form of this relationship depends on some additional assumptions about the form of the detector response, like whether the momentum reconstruction of charge flipped electrons is worse than unflipped electrons (it is), but for now we will consider the simplest case where we assume:

$$\begin{aligned}\lambda^R_{SC}(e_1, e_2) &= \lambda^T(e_1, e_2)\epsilon(e_1)(1-\epsilon(e_2)) + \lambda^T(e_2, e_1)(1-\epsilon(e_1))\epsilon(e_2) \\ &= \lambda^T(e_1, e_2)(\epsilon(e_1) + \epsilon(e_2) - 2\epsilon(e_1)\epsilon(e_2)) =: \lambda^T(e_1, e_2)\epsilon_{12}(e_1, e_2)\end{aligned}$$

and

$$\begin{aligned}\lambda^R_{OC}(e_1, e_2) &= \lambda^T(e_1, e_2)(1-\epsilon(e_1))(1-\epsilon(e_2)) + \lambda^T(e_2, e_1)\epsilon(e_1)\epsilon(e_2) \\ &= \lambda^T(e_1, e_2)(1-\epsilon(e_1) - \epsilon(e_2) + 2\epsilon(e_1)\epsilon(e_2)) = \lambda^T(e_1, e_2)(1-\epsilon_{12}(e_1, e_2)).\end{aligned}$$

In the first expression we introduce $\epsilon_{12}(e_1, e_2)$ as the probability with which a given OC event is reconstructed as an SC event, and similarly in the second

expression the probability an OC event is reconstructed as an OC event is given by $1-\epsilon_{12}(e_1,e_2)$. Notice that under these assumptions, the ratio of these rates is independent of the truth rates,

$$\frac{\lambda^R_{OC}(e_1,e_2)}{\lambda^R_{SC}(e_1,e_2)} = \frac{(1-\epsilon(e_1))(1-\epsilon(e_2))+\epsilon(e_1)\epsilon(e_2)}{\epsilon(e_1)(1-\epsilon(e_2))+(1-\epsilon(e_1))\epsilon(e_2)} \tag{1}$$

$$= \frac{1+\frac{\epsilon(e_1)}{1-\epsilon(e_1)}\frac{\epsilon(e_2)}{1-\epsilon(e_2)}}{\frac{\epsilon(e_1)}{1-\epsilon(e_1)}+\frac{\epsilon(e_2)}{1-\epsilon(e_2)}}. \tag{2}$$

In a given reco di-electron dataset, the distribution of $e_{1/2}$ and their reco charge status can be written as

$$p(OC\wedge(e_1,e_2)) = \frac{\lambda^R_{OC}(e_1,e_2)}{\int de_1 de_2 \lambda^R_{OC}(e_1,e_2)+\int de_1 de_2 \lambda^R_{SC}(e_1,e_2)}$$

and similarly,

$$p(SC\wedge(e_1,e_2)) = \frac{\lambda^R_{SC}(e_1,e_2)}{\int de_1 de_2 \lambda^R_{OC}(e_1,e_2)+\int de_1 de_2 \lambda^R_{SC}(e_1,e_2)}.$$

Therefore the log probability ratio is similarly given by,

$$\begin{aligned}\log\frac{p(OC\wedge(e_1,e_2))}{p(SC\wedge(e_1,e_2))} &= \log\frac{\lambda^R_{OC}(e_1,e_2)}{\lambda^R_{SC}(e_1,e_2)} \\ &= \log\frac{1+\frac{\epsilon(e_1)}{1-\epsilon(e_1)}\frac{\epsilon(e_2)}{1-\epsilon(e_2)}}{\frac{\epsilon(e_1)}{1-\epsilon(e_1)}+\frac{\epsilon(e_2)}{1-\epsilon(e_2)}} \\ &=: \log(1+e^{\kappa(e_1)}e^{\kappa(e_2)})-\log(e^{\kappa(e_1)}+e^{\kappa(e_2)}). \end{aligned} \tag{3}$$

Since neural networks can output any number in $\mathbb{R}$, we introduce $\kappa(e) := \log\frac{\epsilon(e)}{1-\epsilon(e)}$ as a rescaled variable suitable for prediction by a NN. Now this log probability ratio is precisely the function that minimizes the expected value of the binary cross-entropy loss,

$$\mathcal{L}[f] = -\frac{1}{N}\sum_{i=1}^{N} l_i \log s(f(e_1^i,e_2^i)) + (1-l_i)\log(1-s(f(e_1^i,e_2^i))),$$

where an event is labelled $l_i = 1$ if it is SC and 0 if it is OC. A well trained classifier $f_{\min}(e_1,e_2)$ between OC and SC events therefore converges onto an approximation of $\log(1+e^{\kappa(e_1)}e^{\kappa(e_2)})-\log(e^{\kappa(e_1)}+e^{\kappa(e_2)})$.

But we actually want access to the individual $\kappa(e_{1/2})$ values (which can be converted into $\epsilon(e_{1/2})$), but these cannot be extracted directly from $\log(1+e^{\kappa(e_1)}e^{\kappa(e_2)})-\log(e^{\kappa(e_1)}+e^{\kappa(e_2)})$. The trick here is to parametrise the classifier using a network that predicts the κ value of an electron, and to build the log-likelihood ratio out of the predicted κ values using the structure of

the minimum of the loss in Equation 3. In practice this means using a NN, $\phi(e)$, that produces a real number given an electron and to define $f(e_1, e_2) = \log\left(1 + e^{\phi(e_1)+\phi(e_2)}\right) - \log\left(e^{\phi(e_1)} + e^{\phi(e_2)}\right)$ which then gets used in the loss function. Equation 3 proves that this highly constrained form of $f(e_1, e_2)$ can minimize $\mathcal{L}[f]$ by attaining $\phi(e) = \kappa(e)$. Since ϕ is an approximation of κ, we can recover the network's approximation of the single-electron charge-flip probability using $\text{sigmoid}(\phi(e)) = \frac{1}{1+e^{-\phi(e)}} \approx \frac{1}{1+e^{-\kappa(e)}} = \epsilon(e)$.

1.2 Uniqueness

The previous section proved that $\mathcal{L}[f]$ is minimized $\phi(e) = \kappa(e)$. However, just because the loss is minimized by $\phi(e) = \kappa(e)$, does not mean that it is the only value of $\phi(e)$ that minimizes $\mathcal{L}[f]$. This would be a big problem because it would result in our network not learning the function we want! To understand the degeneracy a little more, realise that the log-probability in Equation 3 is simply the log-ratio of the probability that an OC truth event is reconstructed OC vs SC, $\log \frac{\epsilon_{12}(e_1,e_2)}{1-\epsilon_{12}(e_1,e_2)}$. This is what f is gaurenteed to uniquely converge upon, and so for a given di-electron event we can use this quantity to uniquely solve for $\epsilon_{12}(e_1, e_2)$. But for a given di-electron event there are many different values of the per-electron flip probability that results in the same overall event flip rate. By tuning the charge flip probability of one electron up as the other probability is reduced, the overall event flip rate can be held constant. This is the source of the degeneracy.

In fact the relationship required to hold the flip-rate of a given event constant is easy to calculate. Suppose that

$$
\begin{aligned}
f(e_1, e_2) = \log\left(1 + e^{\phi(e_1)+\phi(e_2)}\right) - \log\left(e^{\phi(e_1)} + e^{\phi(e_2)}\right) &= C \\
\implies \frac{1 + e^{\phi(e_1)+\phi(e_2)}}{e^{\phi(e_1)} + e^{\phi(e_2)}} &= e^C \\
\implies 1 + e^{\phi(e_1)}e^{\phi(e_2)} &= e^C(e^{\phi(e_1)} + e^{\phi(e_2)}) \\
\implies (e^{\phi(e_1)} - e^C)(e^{\phi(e_2)} - e^C) &= e^{2C} - 1 \qquad (4) \\
\implies \phi(e_2) &= \log\left(\frac{e^{\phi(e_1)+C} - 1}{e^{\phi(e_1)} - e^C}\right). \qquad (5)
\end{aligned}
$$

So to hold the event flip rate at a constant C, any value of $\phi(e_1) < C$ is a valid solution provided the value of $\phi(e_2)$ is taken as above. At first glance this appears to be a massive problem. Fortunately, once you take into account the fact that the loss contains tens of millions of events, suddenly it seems incredibly improbable that the network converges onto a degenerate solution that satisfies Equation 5 for every pair (e_1, e_2) other than $\phi(e) = \kappa(e)$. Ultimately this isn't a proof, but an argument, and the efficacy will need to be tested in Monte Carlo simulations.

Note that this problem also exists for the existing di-electron based QmisID methods even if its not explicitly discussed.

2 Summary of plan

1. Create a clean dataset of OC/SC di-electron events
2. Train a NN, $\phi(e)$, using as many features as we like of e to minimise the binary cross-entropy loss function $\mathcal{L}[f]$ using the constrained classifier $f(e_1, e_2) = \log\left(1 + e^{\phi(e_1)+\phi(e_2)}\right) - \log\left(e^{\phi(e_1)} + e^{\phi(e_2)}\right)$.
3. Use sigmoid($\phi(e)$) as an approximation for $\epsilon(e)$.
4. Do checks with truth-matched MC.
5. Possibly iterate accounting for backgrounds and Z-peak shift.
6. Apply to data and check results.