JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

Search for Highly Ionizing Particles with the Pixel Detector in the Belle II Experiment - Supporting Information

Suche nach hochionisierenden Teilchen mit dem Pixeldetektor im Belle II Experiment

by
Katharina Dort

supervised by

PD Dr. Sören Lange AR
II. Physics Institute
Faculty 07
Justus-Liebig-Universität Gießen

May 2019

CONTENTS

CHAPTER 1

NEURAL NETWORKS

1.1 Preparation of Input Data

A proper pre-processing of input vectors is considered to be crucial for the performance of machine learning techniques [1]. The normalization or standardization of a data set are two popular pre-processing procedures. Both techniques were tested in order to optimize the application of neural networks for this thesis.

Normalization scales all variables in an input vector to the range [0, 1]:

$$x_{j,\text{norm}} = \frac{x_j - x_{\min}}{x_{\max} - x_{\min}} \tag{1.1.1}$$

where $x_{\min}$ /$x_{\max}$ is the minimal/maximal value of a given dataset and $x_j/x_{j,\text{norm}}$ is the j-th value of this data set before/after normalization. With all input variables scaled to a common range, the dominance of one input feature over the others is prevented. In the context of PXD cluster properties cluster charge values are about one order of magnitude higher than cluster size properties. The impact of charge values on the response of a neural network will therefore overshadow the influence of cluster size values, if no normalization is performed. The disadvantage

of normalization is the compression of the majority of input vectors to a small range, which is particularly harmful if outliers are present.

Standardization scales the input data according to the mean μ and the standard deviation σ of the dataset:

$$x_{j,\text{std}} = \frac{x_j - \mu}{\sigma}. \tag{1.1.2}$$

The new distribution has a mean at zero and unit variance. The standardized data set is unbounded. Consequently, some input features might exert a higher influence than others. This hierarchy of input features can be overcome or even modified to suit certain requirements by assigning weights to the individual features. The neural networks presented in this thesis are trained on standardized input sets.

1.2 Feedforward Neutral Networks

Feedforward Neutral Networks (FNNs) are a form of supervised learning. Neurons are arranged in layers, which are connected among each other. In contrast to recurrent neural networks, the connections do not form loops [2].

The input or sensor neurons, colored in orange, are activated by their environment consisting of external data. Neurons in the hidden layer, displayed in green, are activated by weighted connections to the input neurons. Networks featuring hidden layers are also referred to as Multilayer Perceptron (MLP). The amount of hidden layers can be arbitrarily high. Neurons in each hidden layer are connected to the neurons in the layers before. The last layer, displayed in blue, is referred to as output layer. It delivers the response of the network to an external entity. All connections are unidirectional i.e. the flow of information proceeds only in the direction towards the output layer. The training process of the network is discussed in the following Section.

1.2.1 Activation of Neurons and Training

The response of a single neuron to a stimulus is determined by its *activation function*. Typically non-linear bounded activation functions with a real-valued output are chosen. Popular choices for the activation function are the hyperbolic

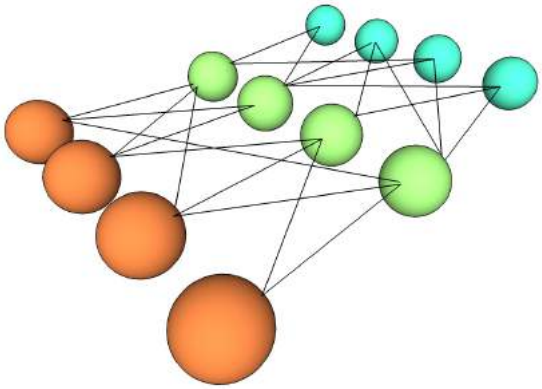

Figure 1.1: Schematic representation of a simple Feedforward Neural Network. The network consists of neurons in the input layer (colored in orange), which are connected to the purple neurons in the hidden layer /green). In turn these neurons share connections to the blue neurons in the output layer. The flow of information proceeds from the input layer, activated by external data, towards the output layer delivering the response of the network.

tangent $t(x_i) = \tanh(x_i)$, the sigmoid function $\sigma(x_i) = 1/(1 + e^{-x_i})$ or a rectifier $r(x_i) = \max(0, x_i)$ where x_i is the input received from a predecessor neuron or external data.

During training/learning the weights between the neurons are adjusted in order to optimize the response of the network. The *back propagation* algorithm is commonly employed for this purpose [3]. For a set of N training vectors $\mathbf{x}_i$ the response $\mathbf{y}_i$ of the neural network is computed. The *disagreement* between $\mathbf{y}_i$ and the desired result $\hat{\mathbf{y}}_\mathbf{i}$ is used to define a so-called *error* or *loss* function E. The geometrical distance between actual and desired output of the network, for instance, can be used as error function:

$$E(\mathbf{x}_1, ..., \mathbf{x}_N) = \frac{1}{2} \sum_{i=1}^{N} (\mathbf{y}_i - \hat{\mathbf{y}}_\mathbf{i}). \tag{1.2.1}$$

The error function is minimized with respect to the weights w_i. A *gradient descent* can be used for this purpose:

$$\mathbf{w}^{(j+1)} = \mathbf{w}^j - l \cdot \nabla_\mathbf{w} E \tag{1.2.2}$$

where $\mathbf{w}^{(j)}$ is the set of weights in the j-th iteration, $\mathbf{w}_j$, l is the learning rate and $\nabla_\mathbf{w} E$ is the derivative of the error function with respect to the set of weights.

1.3 Self-Organizing Maps

Self Organizing Maps (SOMs) apply competitive learning in order to provide a low-dimensional discretized representation of high-dimensional input vectors. Despite the reduction in dimensionality, SOMs preserve the topology of the input to the best degree possible [4].

During training a topologically correct map of the high-dimensional input signal is formed by the lower-dimensional processing units, which are often referred to as *nodes*. A local feedback is responsible for the self-organized mapping process. A SOM is categorized as an unsupervised learning algorithm, since no comparison with a desired output is required.

In the following Section the mathematical concept of SOMs is sketched. A detailed description [5] and a more rigorous mathematical treatment [6] are listed in the bibliography.

1.3.1 Definitions

SOMs consist of two mathematical spaces: a node space of dimension n and a feature space of dimension m. For visualization purposes the dimension of the node space is commonly chosen to be $n = 1$ or $n = 2$. There are N nodes in node space, which sit at a fixed position $r^{(n)} \in \mathbb{R}^n$.

Each node is associated with a feature vector (sometimes referred to as weight vector) of dimension m. The position $r^{(m)} \in \mathbb{R}^m$ of the feature vectors is constantly modified during the learning process as will be explained below.

The input signal consists of vectors $r^{(x)} \in \mathbb{R}^m$. The dimension of the input vectors is therefore equal to the dimension of the weight vectors.

1.3.2 The Learning Algorithm

The neural network is trained by adapting the weight vectors according to the input vectors. For the i-th input vector $r_i^{(x)} = (x_1, x_2, ..., x_m) \in \mathbb{R}^m$ the distance

under a certain metric [1] to all weight vectors $r^{(m)}$ of the map is determined. The *winner vector* $r_w^{(m)}$ is defined as the weight vector with the smallest distance to the input vector:

$$\|r_i^{(x)} - r_w^{(m)}\| = \min_j(\|r_i^{(x)} - r_j^{(m)}\|) \tag{1.3.1}$$

The corresponding node to this weight vector is consequently called *winner node* $r_w^{(n)}$. A local feedback is given to the nodes in a specified region around the winner node. Their associated weight vectors are shifted from their current position $r_j^{(m)}(t)$ to the position $r_j^{(m)}(t+1)$ according to:

$$r_j^{(m)}(t+1) = r_j^{(m)}(t) + h_{w,j}(t)(r_i^{(x)} - r_j^{(m)}(t)) \tag{1.3.2}$$

where the subscript j denotes the j-th weight vector. The parameter t labels each step with an integer number. The initial position of the weight vectors has an impact on the convergence behavior. Nevertheless, this impact is neglected for now and the initial position is chosen randomly.

The function $h_{wj}(t)$ is termed *neighborhood function*. It determines the local relaxation process of the map. The function $h_{wj}(t)$ is monotonically decreasing with increasing time. In addition, with increasing distance from the j-th node to the winner node, the neighborhood function $h_{wj}(t)$ decreases as well. While weight vectors associated with nodes in the near vicinity of the winner node are adapted the most, the weight vectors of distant nodes are not activated. For practical purposes the neighborhood function is chosen such that $0 < h_{wj}(t) < 1$.

Most commonly a Gaussian kernel:

$$h_{wi}(t) = \alpha(t) \exp\left(-\frac{\|r_w^{(n)}(t) - r_i^{(n)}(t)\|^2}{2\sigma^2(t)}\right) \tag{1.3.3}$$

is applied. In the next Section the application of other kernels is briefly discussed. The time-dependent parameter $\alpha(t)$ is considered to be the current *learning rate*. The width of the Gaussian kernel $\sigma(t)$ determines the range of the relaxation process. In order to ensure the monotonic decrease of Eq. 1.3.3 with increasing time, the parameters $\alpha(t)$ and $\sigma(t)$ have to be chosen accordingly.

The choice of $\alpha(t)$ and $\sigma(t)$ is usually guided by intuition as well as *trial and error*.

[1] Even though all metrics are equal in a finite dimensional space, the choice of the metric can influence the convergence behavior.

The expressions:

$$\alpha(t) = \alpha_0(1 - \frac{t}{t_{\max}}) \tag{1.3.4}$$

$$\sigma(t) = \sigma_0(1 - \frac{t}{t_{\max}}) \tag{1.3.5}$$

have turned out to be reasonable choices [7]. Only the constants α_0, σ_0 and $t_{\max}$ remain to be determined for the specific application. In this thesis the constants are fixed to be:

$$\begin{aligned} \alpha_0 &= 0.01 \\ \sigma_0 &= 7 \\ t_{\max} &= 200000 \end{aligned}$$

1.3.3 Alternative Learning Functions

Apart from the Gaussian kernel introduced in the previous Section, two different learning functions are tested in this thesis. The first one is a constant kernel with an extension equal to a width σ, the other one resembles a 'Mexican hat' and is given by:

$$h(t) = (1 - \frac{2p}{2\pi\sigma^2})e^{-\frac{p}{2\pi\sigma^2}} \tag{1.3.6}$$

where p is given by:

$$p = \|r_{w,x}^{(n)}(t) - r_{i,x}^{(n)}(t)\|^2 + \|r_{w,y}^{(n)}(t) - r_{i,y}^{(n)}(t)\|^2 \tag{1.3.7}$$

where the second index indicates the spatial dimension in node space. The Gaussian kernel turned out to deliver the best results. It is chosen as neighborhood function for all SOMs presented in this thesis based on this empirical observation.

CHAPTER 2

COMPOSITION OF INPUT VECTOR FOR NEURAL NETWORKS

2.1 Image Moment Analysis

Visual pattern recognition on the basis of *image moment invariants* has proven to be a valuable tool in computer vision and image analysis [8]. In the following Section the application of image moments for the characterization of PXD clusters is discussed.

Zernike moments [9] have proven to be a successful technique for the classification of shower shapes in calorimeters [10]. Zernike moments are invariant under rotation but depend on the scaling and translation of objects. A more general scheme for constructing a complete set of independent image invariants was first introduced by Flusser [11, 12]. Image moments invariant under scaling, rotation, skewness, kurtosis etc. can be derived from his theorems. There is no universal answer as to which image invariants should be used for a cluster analysis. In some cases, for instance, the rotation of a cluster with respect to the global z-axis can contain valuable information, which are lost, if rotation invariants are used to describe the

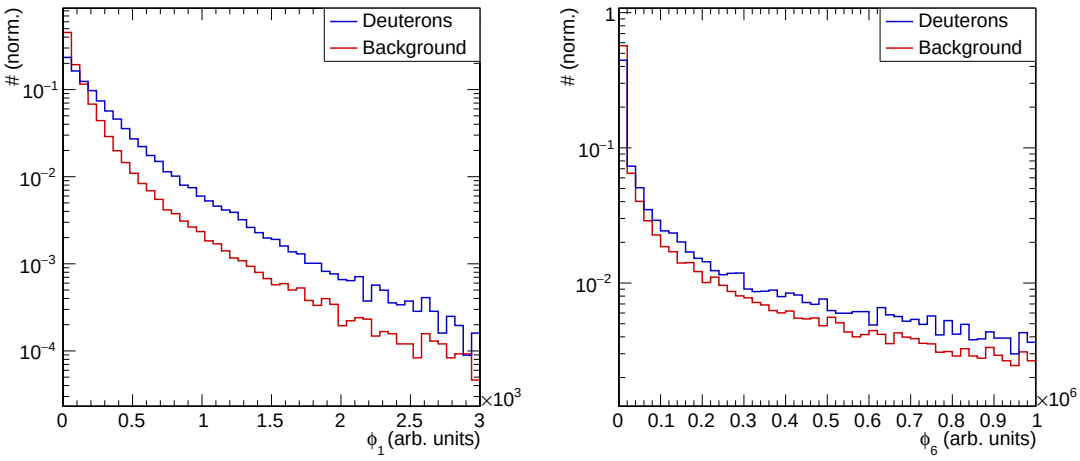

Figure 2.1: Image moments ϕ_1 (left-hand side) and ϕ_6 (right-hand side) for anti-deuterons and Phase III background. While the distributions for ϕ_1 are distinguishable, the variable ϕ_6 has no discriminatory power.

cluster. A pre-evaluation based on physical considerations and prior experimental evidence of cluster properties for a specific analysis is required. In this thesis results for image moments invariant under rotation and translation are shown. The choice is motivated by the desire to keep the well-established application of Zernike moments, which are also invariant under rotation, while extending them to achieve independence from spacial translations.

Image moments are constructed from the complex moments c_{pq}:

$$c_{pq} = \sum_x \sum_y (x + iy)^p (x - iy)^q f(x, y). \tag{2.1.1}$$

where c_{pq} is the image moment of order $(p+q)$. The sum runs over the set of input pixels (x, y). The charge of each pixel is encoded in the function $f(x, y)$.

According the Flussner, a complete set of independent rotation/translation invari-

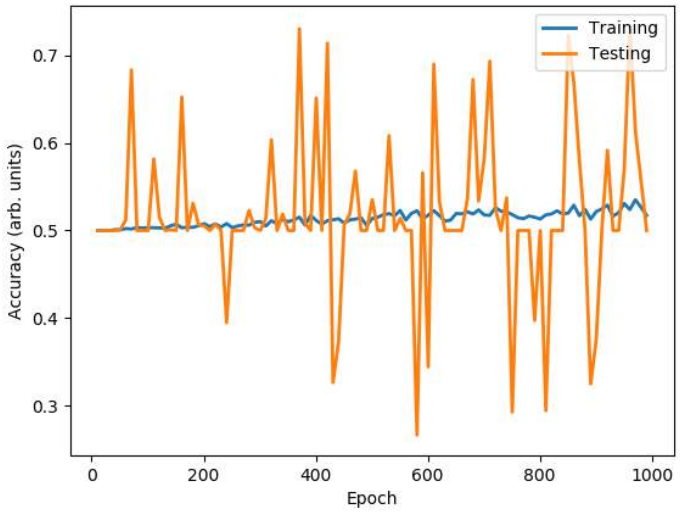

Figure 2.2: Accuracy against training epoch for training and testing set. The input vectors consist of the seven image moments presented in this Section. A cut at 0.5 on the classification axis is set to determine the accuracy. The accuracy for the training set is increasing slightly. The accuracy for the testing set oscillates between ∼0.7 and ∼0.3.

ants are given by:

$$\begin{aligned}
\phi_1 &= c_{11} \\
\phi_2 &= c_{20}c_{02} \\
\phi_3 &= c_{30}c_{03} \\
\phi_4 &= c_{21}c_{12} \\
\phi_5 &= \mathrm{Re}(c_{30}c_{12}^3) \\
\phi_6 &= \mathrm{Re}(c_{20}c_{12}^2) \\
\phi_7 &= \mathrm{Im}(c_{30}c_{12}^3)
\end{aligned} \tag{2.1.2}$$

In Fig. 2.1 the image moments ϕ_1 and ϕ_6 for anti-deuterons and Phase III background are shown. While the variable ϕ_1 exhibits some discriminatory power, the ϕ_6 distribution for anti-deuterons and background is indistinguishable. Distributions for the the entire set of invariants considered for this thesis is shown in Fig. 2.3 at the end of this Section. The application of image moments turned out to perform significantly worse than the other types of input vectors. The distributions in Fig. 2.3 demonstrate that all of the seven invariants are very similarly for

anti-deuterons and background. In Fig. 2.2 the accuracy as a function of training epoch for a training and testing set is depicted. The input vectors consist of the seven image moments. A cut at 0.5 on the classification axis is set. While the accuracy for the training set increases slightly, it scarcely surpasses 0.5. The accuracy for the testing set oscillates between $\sim$0.7 and $\sim$0.3.

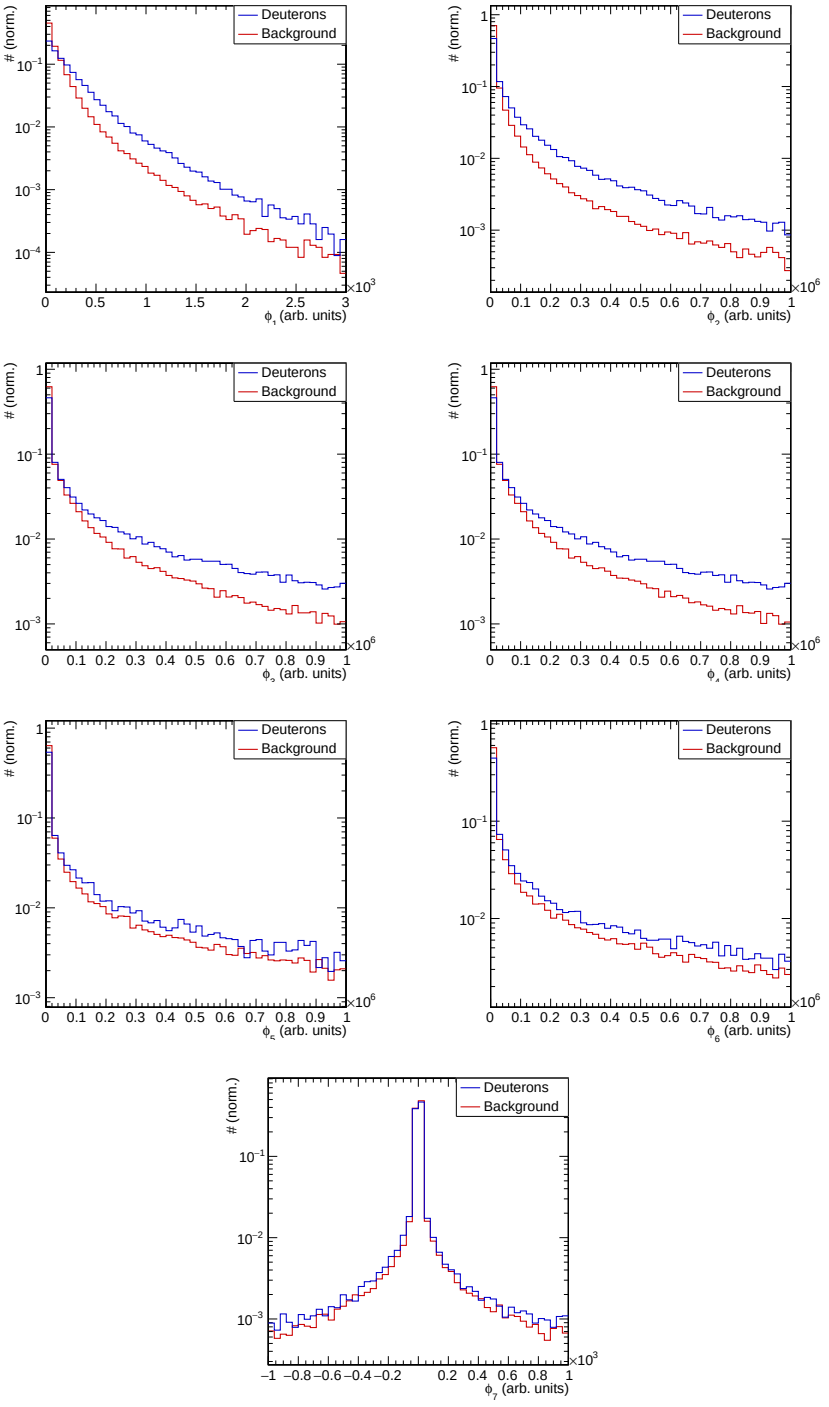

Figure 2.3: Entire set of image moments considered for this thesis. The distributions for each image moment for anti-deuteron and background clusters is shown.

2.1.1 Smallest Enclosing Ellipse

The assignment of a *smallest enclosing ellipse* to clusters allows for the extraction of features such as cluster length, eccentricity, skewness etc., which might serve as input for neural networks.

There are several algorithms to obtain an ellipse, which encloses a set of data points [13, 14]. In the following the *Direct least square fit* [15] is introduced as an example.

Conic sections are defined as a set of points $\hat{\mathbf{x}}$ satisfying the equation:

$$F(\mathbf{a}, \hat{\mathbf{x}}) = D \cdot \mathbf{a} = 0 \tag{2.1.3}$$

where $\mathbf{a}$ and D are defined as $\mathbf{a} = (a_{x,x}\, a_{x,y}\, a_{y,y}\, a_x\, a_y\, a_1)^T$ and $\mathbf{D} = (x^2\, xy\, y^2\, x\, y\, 1)$ for data points $\mathbf{x}_i = (x, y)_i$ in $\hat{\mathbf{x}}$ [16]. The function $F(\mathbf{a}, \hat{\mathbf{x}})$ is also referred to as *algebraic distance*. The fitting of a general conic to N data point $\mathbf{x}_i$ $i = 1, ..., n$ is approached by minimizing the square of the algebraic distances:

$$\Delta(\mathbf{a}, \hat{\mathbf{x}}) = \sum_{i=1}^{N} F(\mathbf{a}, \mathbf{x}_i)^2. \tag{2.1.4}$$

It can be shown [17], that this condition is equivalent to:

$$\Delta(\mathbf{a}, \hat{\mathbf{x}}) = \sum_{i=1}^{N} \mathbf{a}^T \mathbf{D}_i^T \mathbf{D}_i \mathbf{a} = \mathbf{a}^T S \mathbf{a} \tag{2.1.5}$$

where $S = \sum_{i=1}^{N} \mathbf{D}_i^T \mathbf{D}_i$ is a $6 \,\mathrm{x}\, 6$ matrix.

In the special case of an ellipse the vector $\mathbf{a}$ has to satisfy the condition

$$4 a_{x,x} a_{y,y} - a_{x,y}^2 > 0 \tag{2.1.6}$$

which can be written as matrix equation [15, 18] by introducing the $6 \,\mathrm{x}\, 6$ matrix C, with all entries equal to zero except for $C_{1,3} = C_{3,1} = 2$ and $C_{2,2} = -1$:

$$\mathbf{a}^T C \mathbf{a} > 0 \tag{2.1.7}$$

$$\mathbf{a}^T C \mathbf{a} = \phi \tag{2.1.8}$$

where ϕ is a positive number. The last condition holds, since $F(\mathbf{a}, \hat{\mathbf{x}}) = 0$ is independent of scaling in $\mathbf{a}$.

Consequently, the vector $\mathbf{a}$ has to be chosen such that $\Delta(\mathbf{a}, \hat{\mathbf{x}})$ is minimal under the constraint $\mathbf{a}^T C\mathbf{a} = \phi$. In terms of *Lagrange Multipliers* this is written as:

$$L(\mathbf{a}) = \Delta(\mathbf{a}, \hat{\mathbf{x}}) - \lambda(\mathbf{a}^T C\mathbf{a} - \phi) \tag{2.1.9}$$

where $L(\mathbf{a})$ is the Lagrange function and λ is a Lagrange multiplier. Setting $\partial_{\mathbf{a}} L(\mathbf{a}) = 0$ one obtains the eigenvalue equation:

$$\frac{1}{\lambda}\mathbf{a} = S^{-1}C\mathbf{a}. \tag{2.1.10}$$

The pixel positions belonging to a cluster $\mathbf{x}_i = (x, y)_i$ are used as input to the eigenvalue equation to obtain the smallest ellipse around the cluster.

Ellipse parameters

The fit of the smallest enclosing ellipse yields the vector $\mathbf{a}$. The following section demonstrates how the properties of the fitted ellipse are inferred from $\mathbf{a}$.

The angle of rotation α with respect to the x coordinate is given by

$$\alpha = \frac{1}{2}\arctan\left(\frac{a_{x,y}}{a_{x,x} - a_{y,y}}\right). \tag{2.1.11}$$

In order to express the length of the major and minor axis of the ellipse in terms of $\mathbf{a}$, the following substitutions are used: a, b, c, d, e, f, $g = a_{x,x}$, $a_{x,y}/2$, $a_{y,y}$, $a_x/2$, $a_y/2$, a_1 The length of the axes are defined as:

$$l_{\text{major}} = \sqrt{\frac{2(af^2 + cd^2 + gb^2 - 2bdf - acg)}{(b^2 - ac)(\sqrt{(a-c)^2 + 4b^2} - (a+c))}} \tag{2.1.12}$$

$$l_{\text{minor}} = \sqrt{\frac{2(af^2 + cd^2 + gb^2 - 2bdf - acg)}{(b^2 - ac)(-\sqrt{(a-c)^2 + 4b^2} - (a+c))}} \tag{2.1.13}$$

The eccentricity ϵ is obtained by computing:

$$\epsilon = \sqrt{1 - \frac{b^2}{a^2}}. \tag{2.1.14}$$

In Fig. 2.4 examples of clusters fitted with smallest enclosing ellipses are shown. The x coordinate in the above equations corresponds to the local v coordinate and

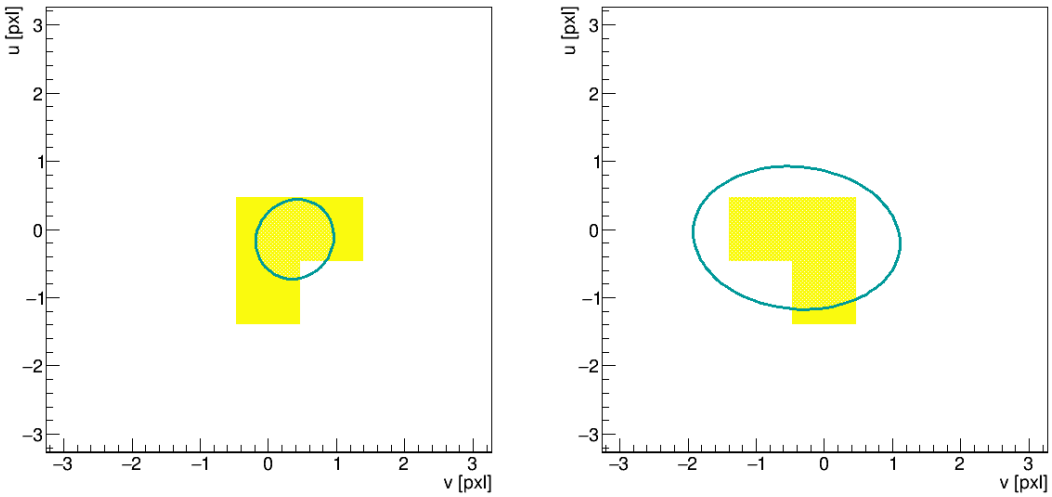

Figure 2.4: Smallest enclosing ellipse (cyan line) fitted to PXD clusters with size three. The ellipses do not capture the entire extend of the cluster.

y to the u coordinate in the local PXD coordinate system. The center of the cluster is shifted to $(u, v) = (0, 0)$. A smallest enclosing ellipse is expected to preserve the orientation of the cluster in the uv plane. Additionally, all pixels belonging to the clusters should be located within the ellipse. The second condition is not fulfilled by the two ellipses shown in Fig. 2.4. The spatial extend of the pixels is not taken into account since the fit routine only receives information about the center of the pixels. The issue could be remedied by including information about the boundaries of the cluster. This would require pre-processing of every single cluster prolonging the computation.

The first condition can be compromised as well for large clusters as shown in Fig. 2.5. Additionally, the implementation of the algorithm turned out to be numerically unstable due to the computation of the inverse for Eq. 2.1.10. In the worst cases the fit does converge, but the agreement between data and ellipse is insufficient (see Fig. 2.5). An additional algorithm to detect the goodness of the fit is required to prevent a mismatch between cluster and fitted ellipse. An increase of the computational difficulty and expense is the consequence. There are, however, suggestions how the numerical stability of the least square fitting of ellipses can be improved [19].

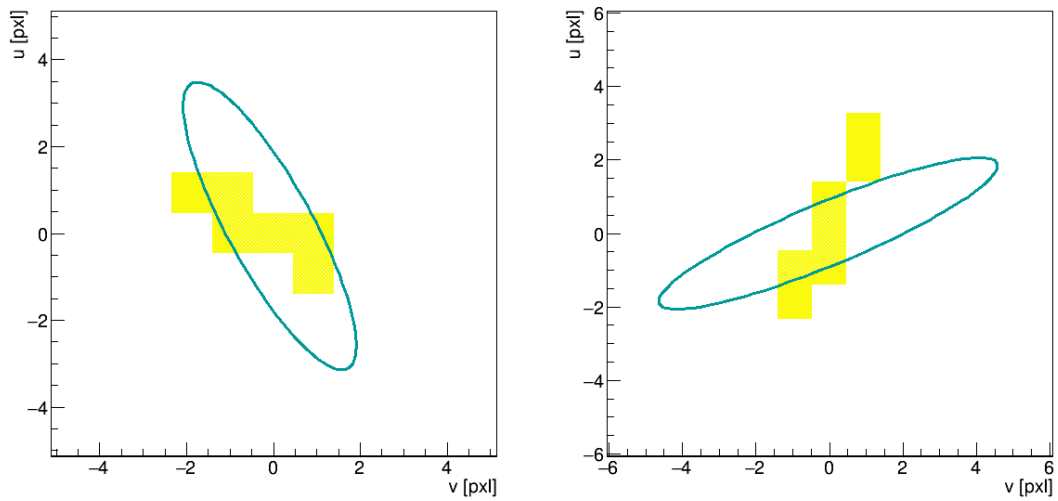

Figure 2.5: Smallest enclosing ellipse (cyan line) fitted to PXD clusters with size six (left-hand side) and size seven (right-hand side). The ellipses do not adopt the orientation of the cluster.

2.1.2 Cluster Angle - Principal Component Analysis

A Principal Component Analysis (PCA) [20,21] is performed in order to determine the angle between a cluster and the global z axis.

The covariance matrix C of a cluster is given by:

$$C = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) \end{pmatrix} \tag{2.1.15}$$

where the covariance $\mathrm{cov}(x,y)$ is determined by:

$$\mathrm{cov}(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}. \tag{2.1.16}$$

n is the amount of pixels belonging to the cluster. For the i-th pixel, the x coordinate x_i and the y coordinate y_i are subtracted from the respective mean, which is obtained by computing:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}. \tag{2.1.17}$$

Note that, $\mathrm{cov}(x,x)$ as well as $\mathrm{cov}(y,y)$ are per definition equal to the variance. For a cluster with size $n = 1$ the PCA is not applied to avoid division by zero (see Eq. 2.1.16).

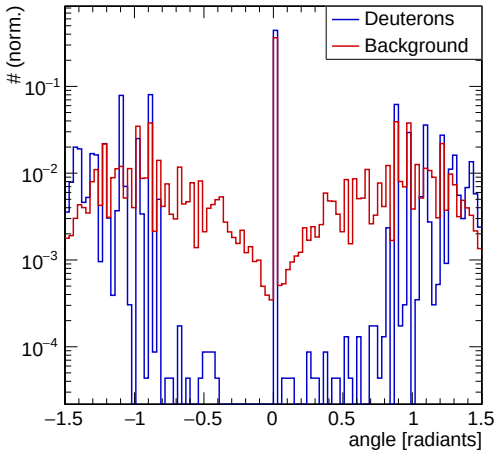

Figure 2.6: Cluster angle for anti-deuterons with $p < 3\,\text{GeV}$ and background. Both distributions exhibit a maximum at 0, which corresponds to clusters aligned with the v axis. The occurrence of clusters with a non-zero angle is more common for background than for anti-deuterons.

The principle components λ_1 and λ_2 are defined as the eigenvalues of the matrix 2.1.15. The eigenvector $\mathbf{a}(\lambda_1)$ with the largest eigenvalue λ_1 is associated with the major axis of a cluster. Accordingly the eigenvector with the lower eigenvalue $\mathbf{a}(\lambda_2)$ represents the minor axis. The orientation of a cluster is defined as:

$$\alpha = \arccos\left(\frac{\mathbf{a}_x(\lambda_1)}{\sqrt{\mathbf{a}_x(\lambda_1)^2 + \mathbf{a}_y(\lambda_1)^2}}\right). \tag{2.1.18}$$

As an alternative to calculating the geometrical angle of a cluster, the inputs to the PCA could be weighted with the pixel charge yielding a charge-weighted angle.

In Fig. 2.6 the cluster angle for anti-deuterons with momentum below 3 GeV and background is depicted. Both distributions adopt a maximum at 0, which is equivalent to a cluster parallel to the v/z axis. The occurrence of clusters, which are unaligned with the v/z axis is strongly suppressed for both particles species. Anti-deuteron clusters with non-zero angles, however, are more infrequent than background clusters in accordance with the higher cluster size in u direction for background clusters. The geometrical cluster angle can be partially reconstructed from the cluster size in u and v direction. With increasing cluster size in u the cluster angle approaches $\pi/2$ for a given size in v. The reason for the higher number of clusters with non-zero angle for background is therefore equivalent to the

higher size in u direction: the off-vertex source of beam background particles.

The calculation of the cluster angle turned out to be numerically more stable than the computation of the smallest enclosing ellipse. However, the computation is time consuming (determination of eigenvalues) and does not yield a high amount of new information. The distribution of cluster angles is found to be partially describable with the cluster size in u and v. The cluster angle is therefore not added as property in the input vector to the neural networks.

BIBLIOGRAPHY

[1] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Data preprocessing for supervised leaning," *International Journal of Computer Science*, vol. 1, no. 2, pp. 111–117, 2006.

[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[3] A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss, M. Backes, T. Carli, O. Cohen, A. Christov, *et al.*, "TMVA-Toolkit for multivariate data analysis," *arXiv preprint physics/0703039*, 2007.

[4] K. Kiviluoto, "Topology preservation in self-organizing maps," in *Neural Networks, 1996., IEEE International Conference on*, vol. 1, pp. 294–299, IEEE, 1996.

[5] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[6] M. Cottrell, J.-C. Fort, and G. Pagès, "Theoretical aspects of the SOM algorithm," *Neurocomputing*, vol. 21, no. 1-3, pp. 119–138, 1998.

[7] F. Mulier and V. Cherkassky, "Learning rate schedules for self-organizing maps," in *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, vol. 2, pp. 224–228, IEEE, 1994.

[8] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962.

[9] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 5, pp. 489–497, 1990.

[10] R. Sinkus and T. Voss, "Particle identification with neural networks using a rotational invariant moment representation," *Nuclear Instruments and Methods in Physics Research Section A*, vol. 391, no. 2, pp. 360–368, 1997.

[11] J. Flusser, "On the independence of rotation moment invariants," *Pattern recognition*, vol. 33, no. 9, pp. 1405–1410, 2000.

[12] J. Flusser, T. Suk, and B. Zitova, "2d and 3D Image Analysis by Moments," 2016.

[13] C. F. Van Loan, "Using the Ellipse to Fit and Enclose Data Points," *Department of Computer Science Cornell University*, p. 54, 2008.

[14] B. Gärtner and S. Schönherr, "Exact primitives for smallest enclosing ellipses," *Information Processing Letters*, vol. 68, no. 1, pp. 33–38, 1998.

[15] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 5, pp. 476–480, 1999.

[16] "Mathematica Quadratic Curve." `http://mathworld.wolfram.com/QuadraticCurve.html/`. Accessed: 2018-11-19.

[17] F. L. Bookstein, "Fitting conic sections to scattered data," *Computer Graphics and Image Processing*, vol. 9, no. 1, pp. 56–71, 1979.

[18] A. W. Fitzgibbon, R. B. Fisher, *et al.*, "A buyer's guide to conic fitting," *DAI Research paper*, 1996.

[19] R. Halır and J. Flusser, "Numerically stable direct least squares fitting of ellipses," in *Proc. 6th International Conference in Central Europe on Computer Graphics and Visualization. WSCG*, vol. 98, pp. 125–132, Citeseer, 1998.

[20] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[21] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.