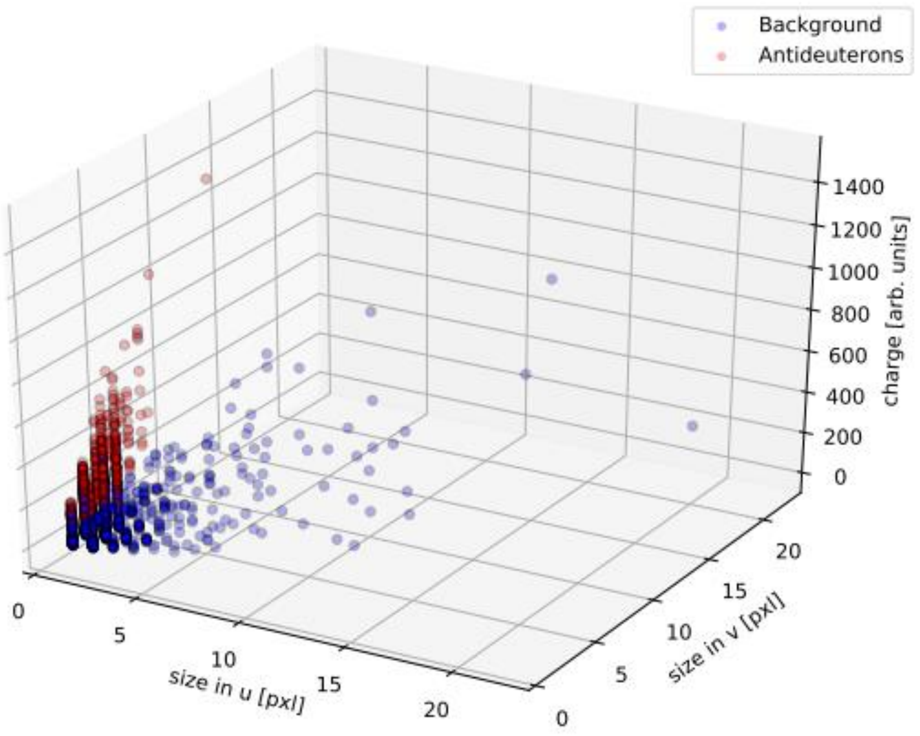
Using Support Vector Machines (SVMs) to find Slow Pions in the PXD

Timo Schellhaas | Belle II Slow Pion Tracking Workshop | JLU Gießen | Garching 10.11.2021

Content

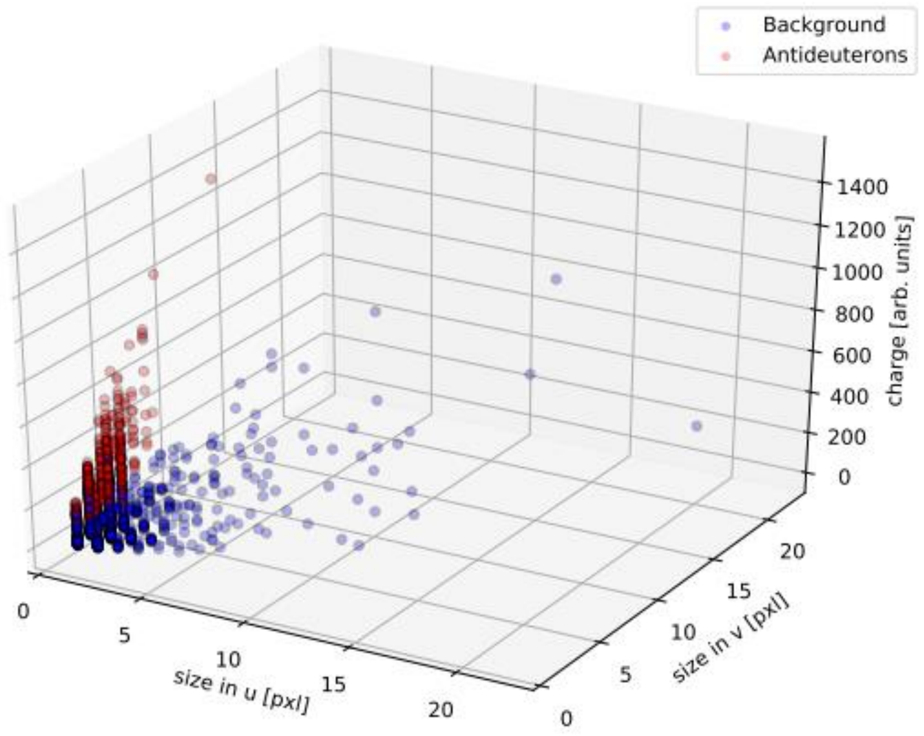
- Why SVMs?
- How do SVMs work?
- How to improve the SVM algorithm?
- Example plot of the slow pions data
- Dataset
- Results
- Summary

Why SVMs?



S. Käs, Multiparameter Analysis of the Belle II Pixeldetector's Data

Why SVMs?

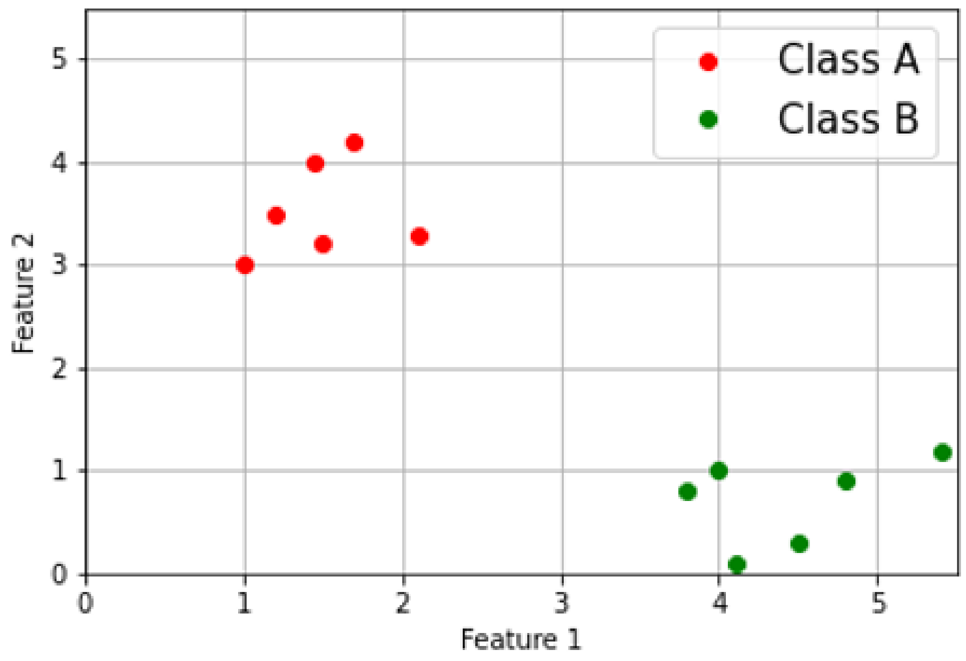


S. Käs, Multiparameter Analysis of the Belle II Pixeldetector's Data.

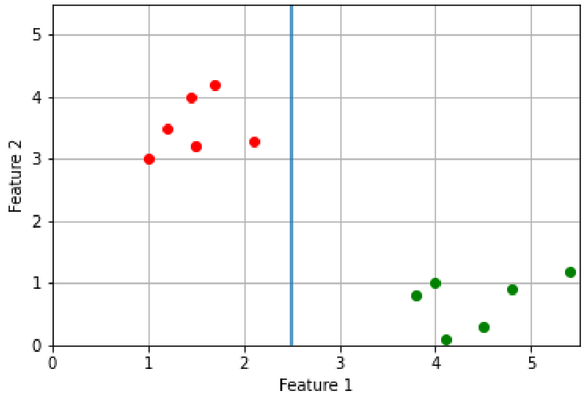
Separable in higher dimension?

How do SVMs work?

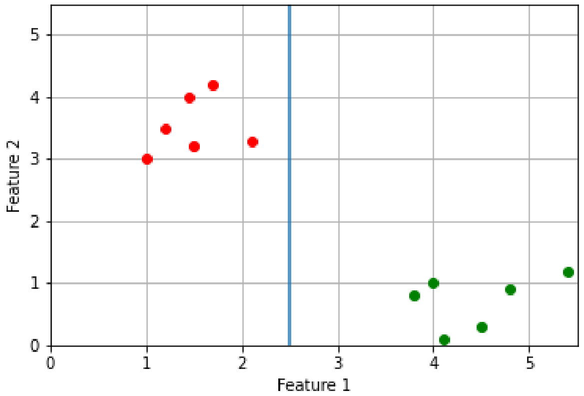
- Algorithm for classification
- Idea: separate classes with a line
- Note: Also possible for higher dimensions

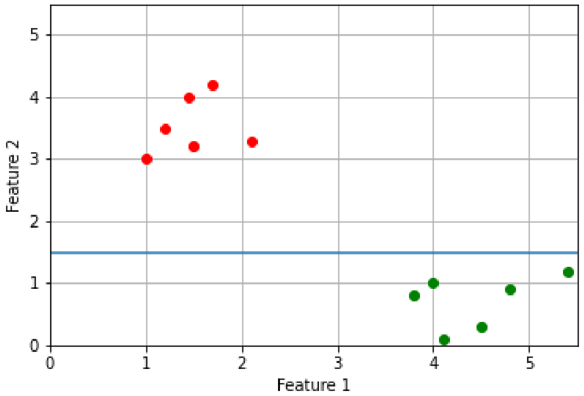


How do SVMs work?

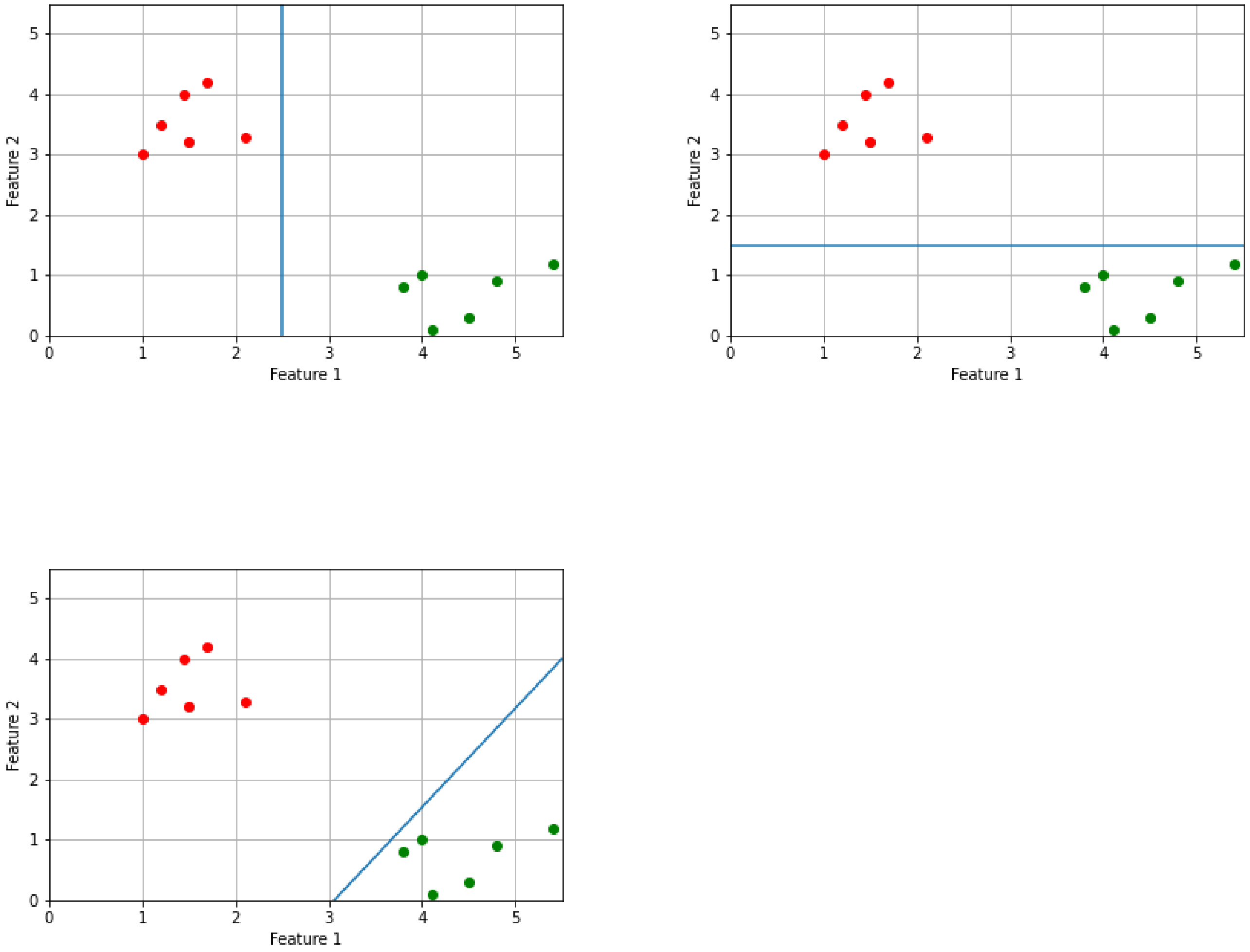


How do SVMs work?

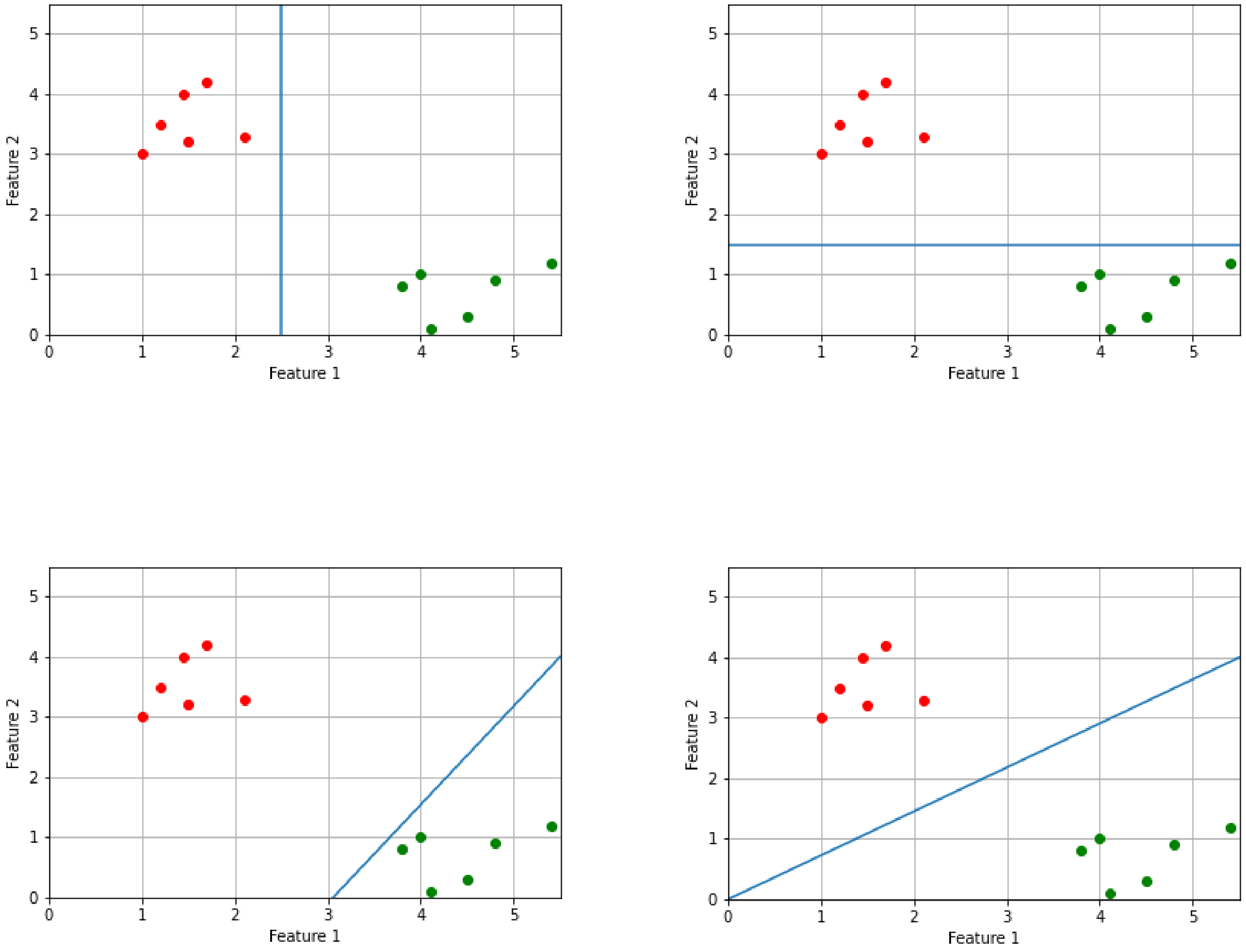




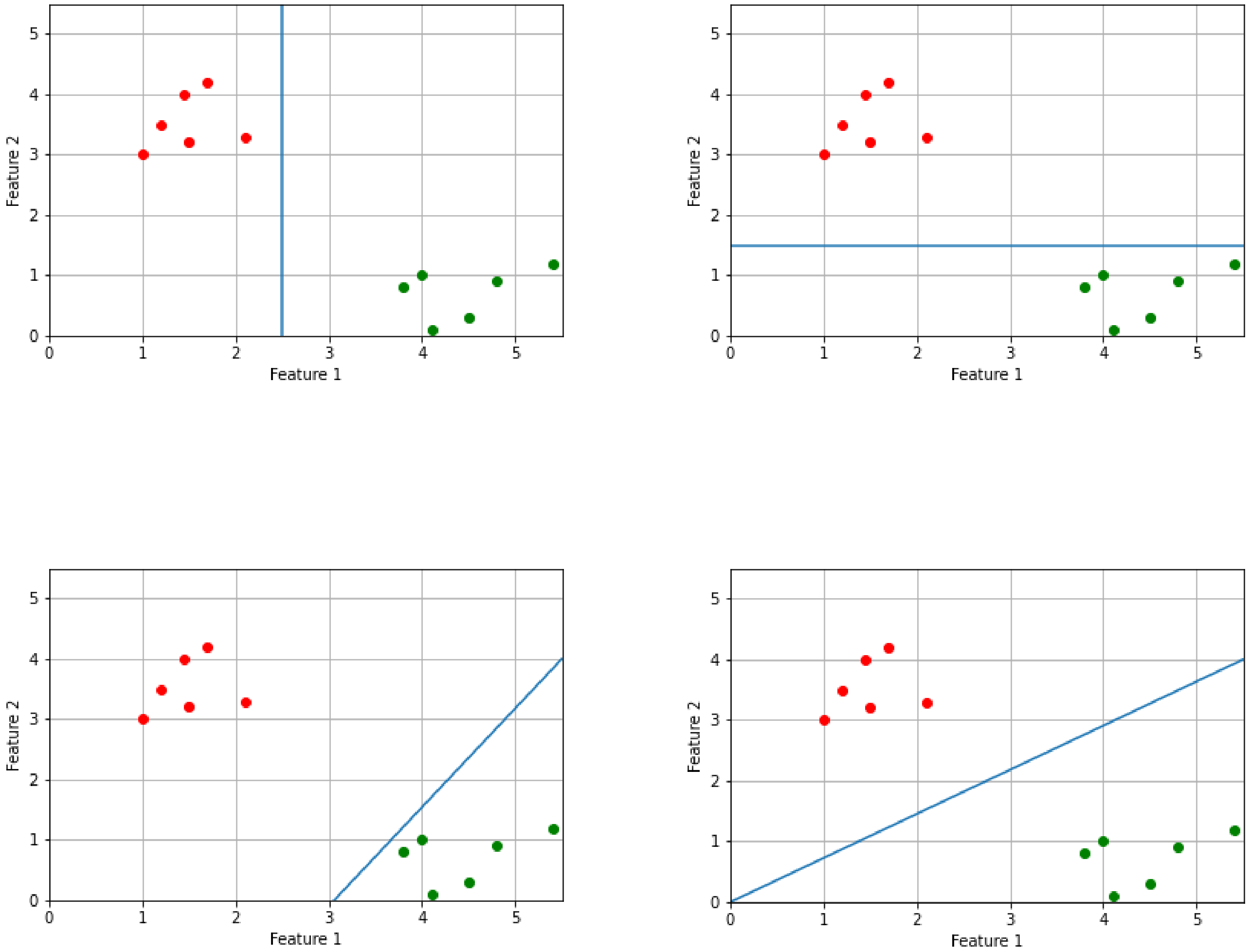
How do SVMs work?



How do SVMs work?

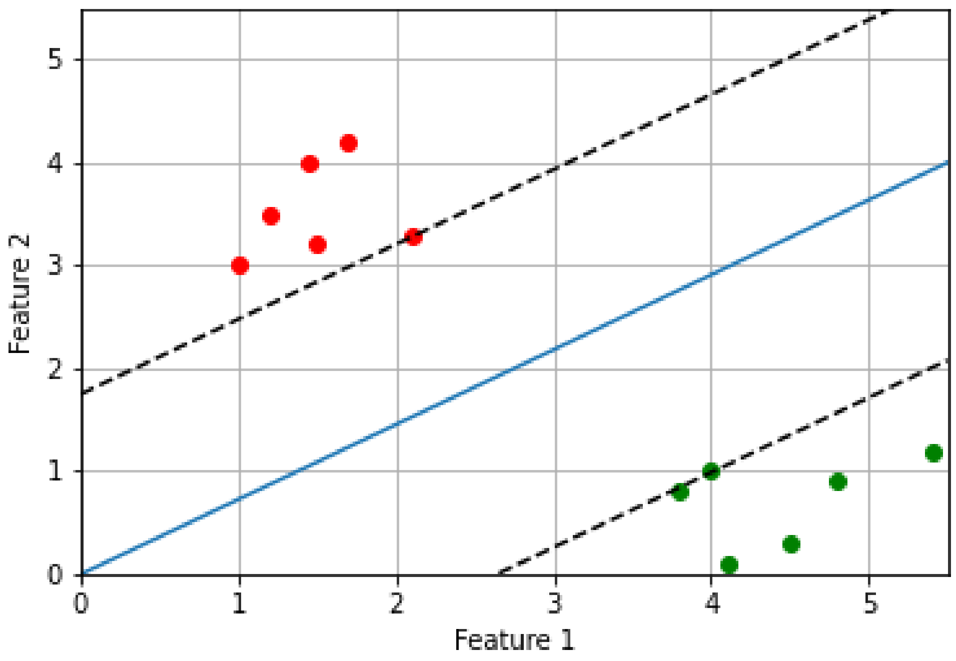


How do SVMs work?



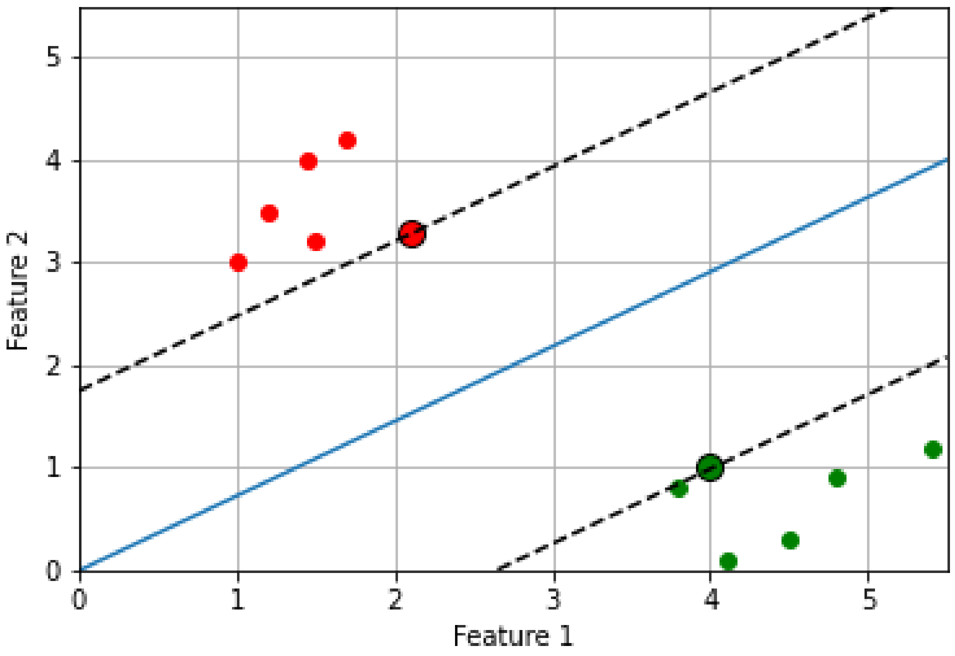
How do SVMs work?

- Take line with the maximum margin
- "Optimal Margin Classifier"



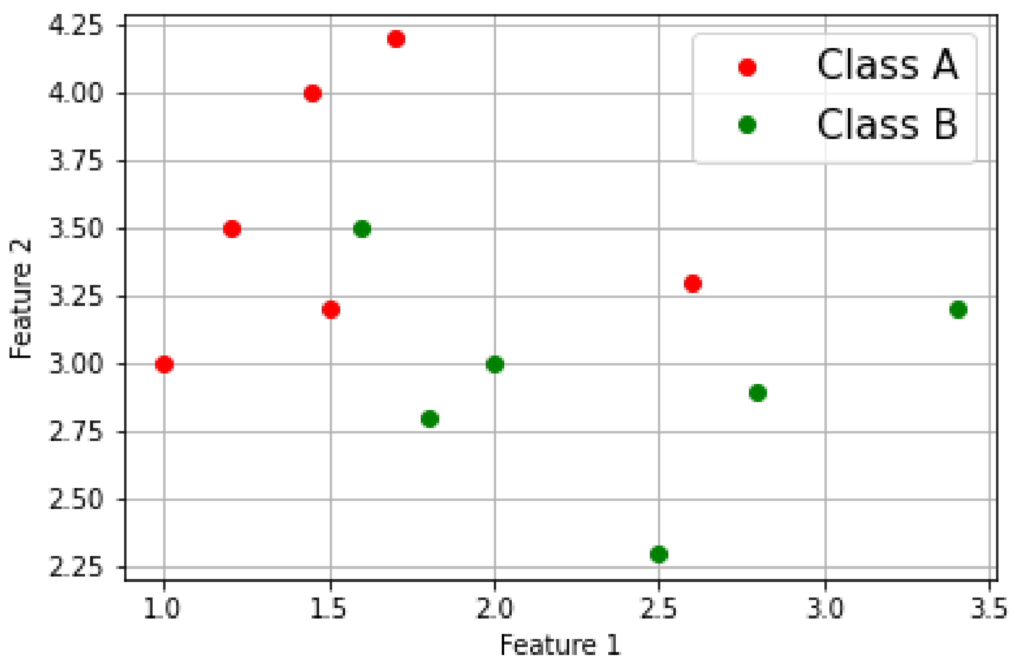
How do SVMs work?

Support Vectors



How to improve the SVM algorithm?

- Problem: not every dataset can be separated by a line

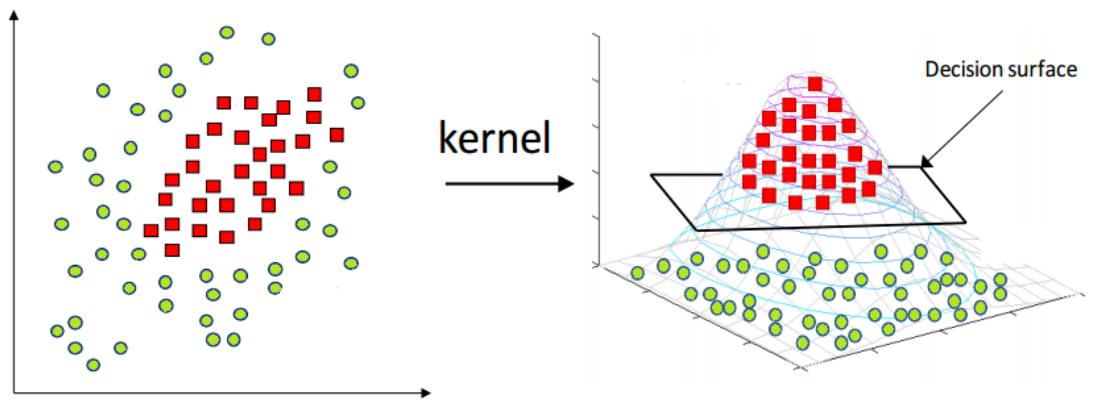


How to improve the SVM algorithm?

- Idea: transform 2-dim feature vector into 3-dim vector

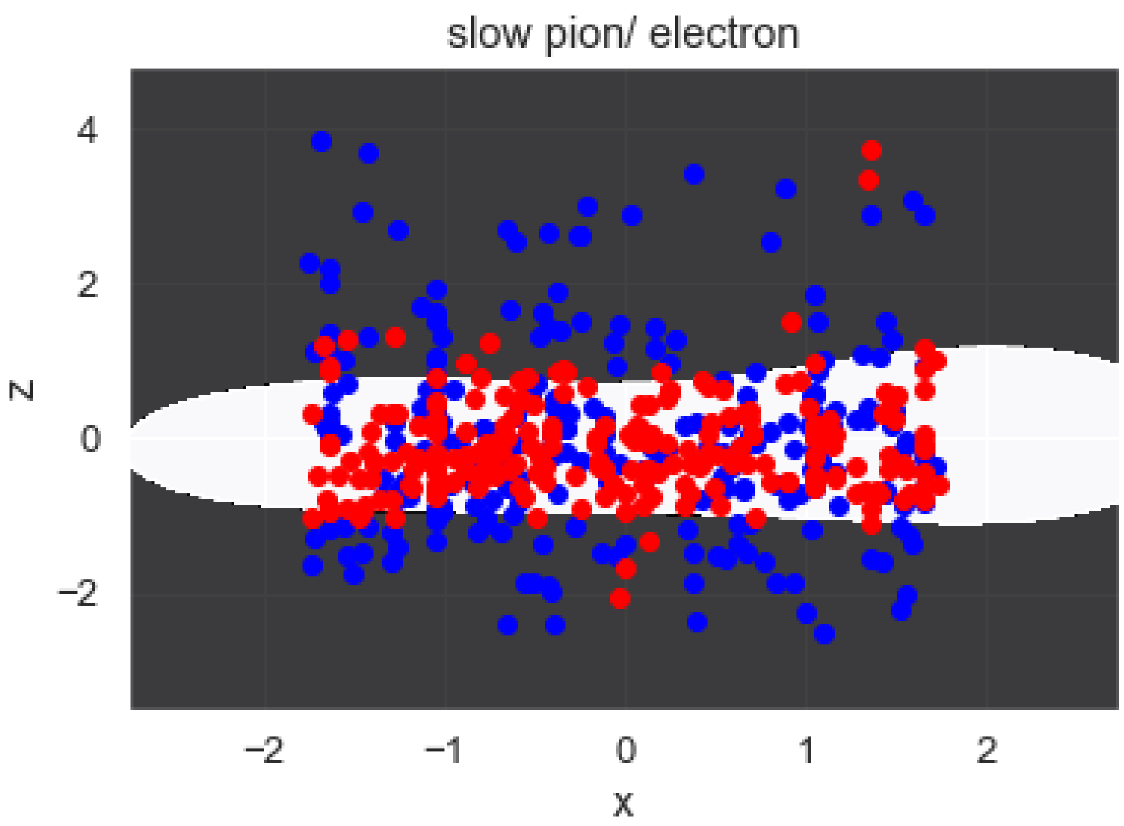
$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

- Separate classes using a hyperplane



Source: Simple Tutorial on SVM and Parameter Tuning in Python and R | HackerEarth Blog

Example plot of slow pions dataset

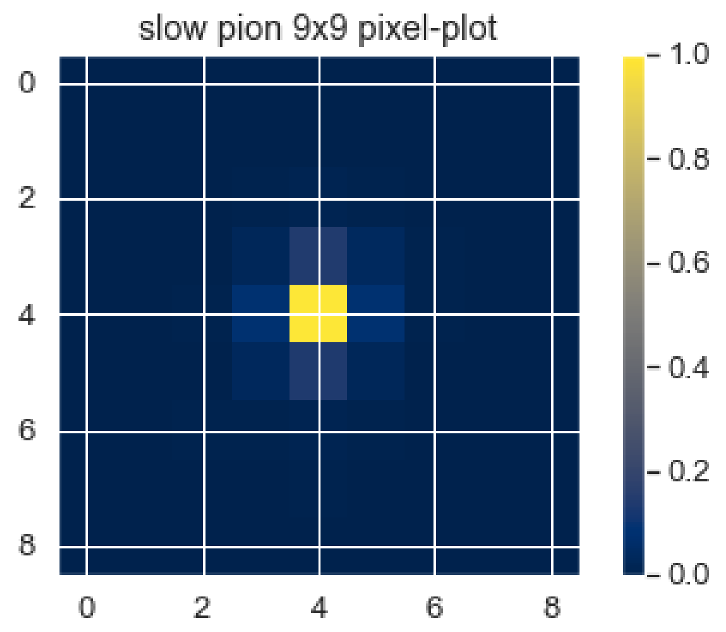


Red points: electrons
Blue points: slow pions
White area: classified as electron
Black area: classified as slow pion

Dataset: Slow Pions/ Electrons

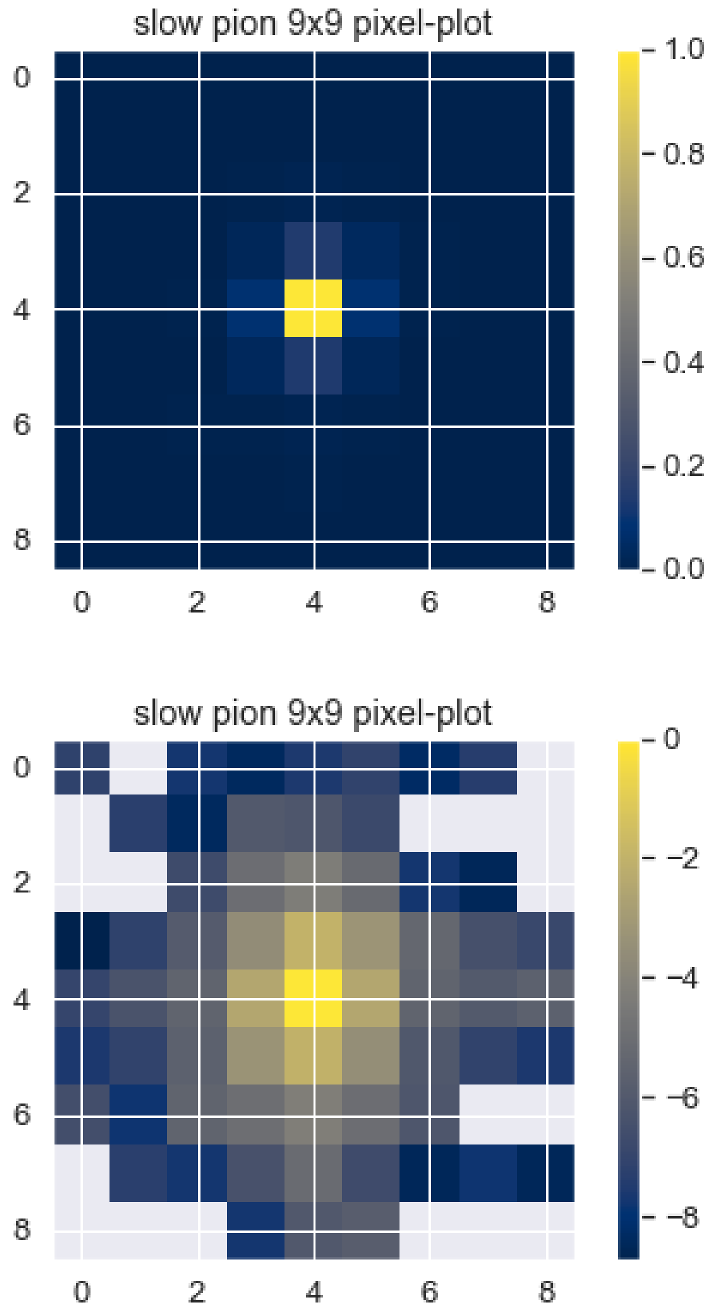
- Total Cluster Charge
- 9x9 Matrix
- x, y, z position

Dataset: Slow Pions/ Electrons



Dataset: Slow Pions/ Electrons

logarithmic scale



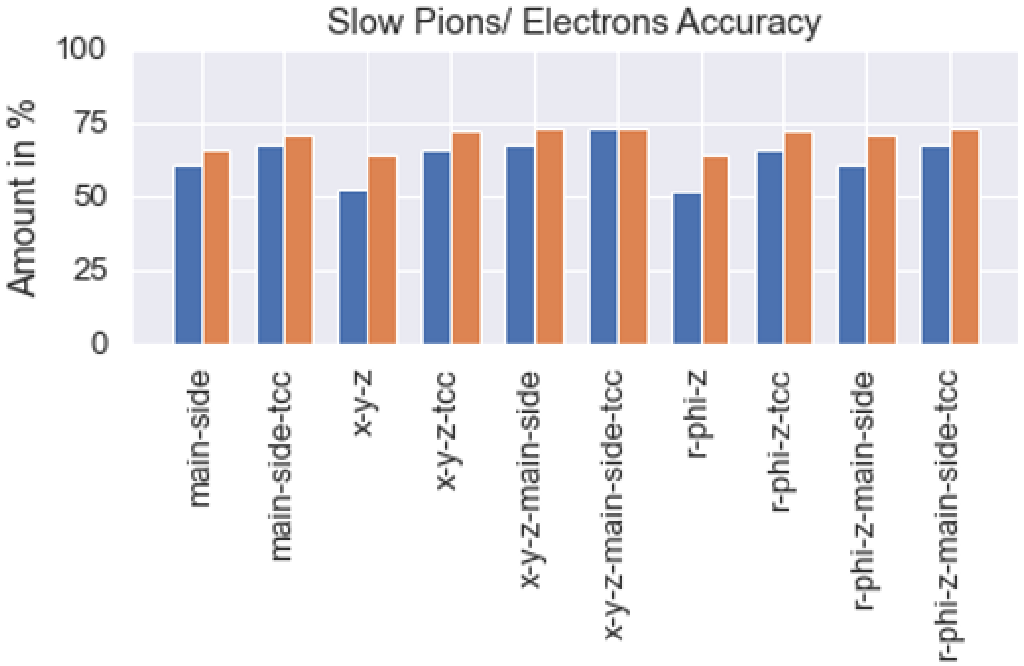
Results

- Applied to slow pions/ electrons with a linear kernel and rbf kernel
- $K(x,y) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - y_{ij})^2)$
- Calculate accuracy, purity, efficiency and rejection

Results: Accuracy

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

"amount of correctly classified particles"



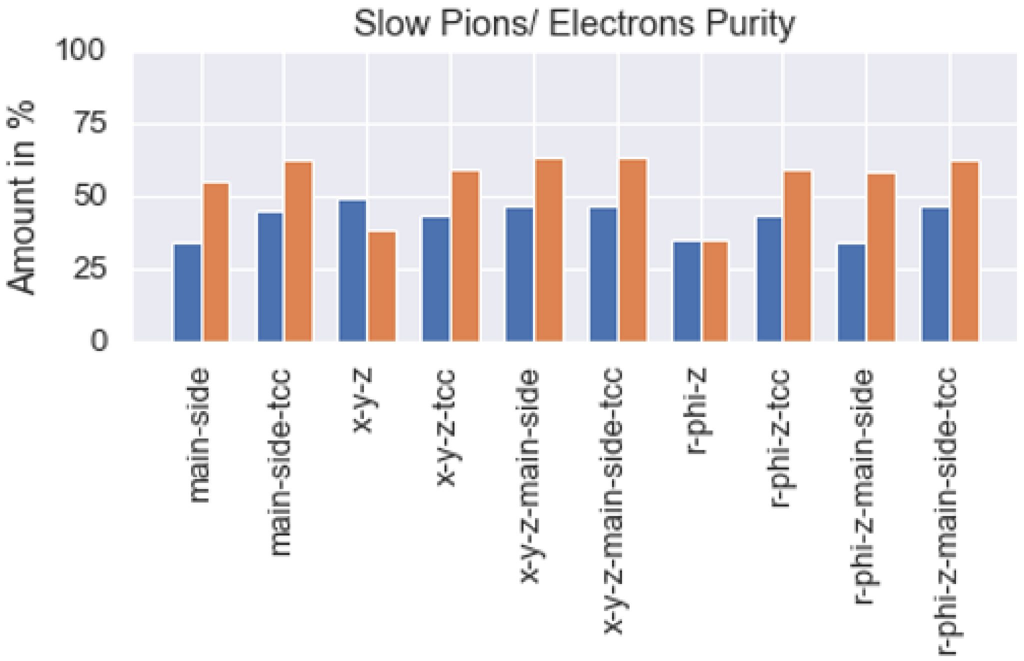
orange: rbf
blue: linear

Best: x-y-z-main-side-tcc

Results: Purity (Sensitivity)

$$PPV = \frac{TP}{TP+FN}$$

"amount of correctly classified pions among all pions"



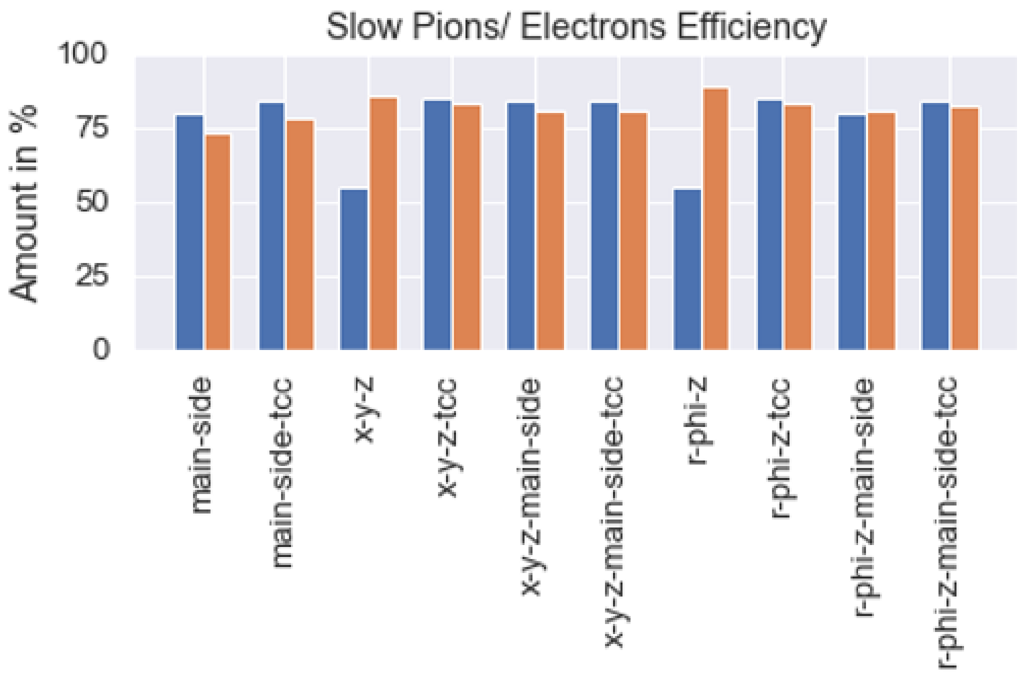
orange: rbf
blue: linear

Best: x-y-z-main-side-tcc

Results: Efficiency (Precision)

$$TPR = \frac{TP}{TP + FP}$$

"amount of pions among particles classified as pions"



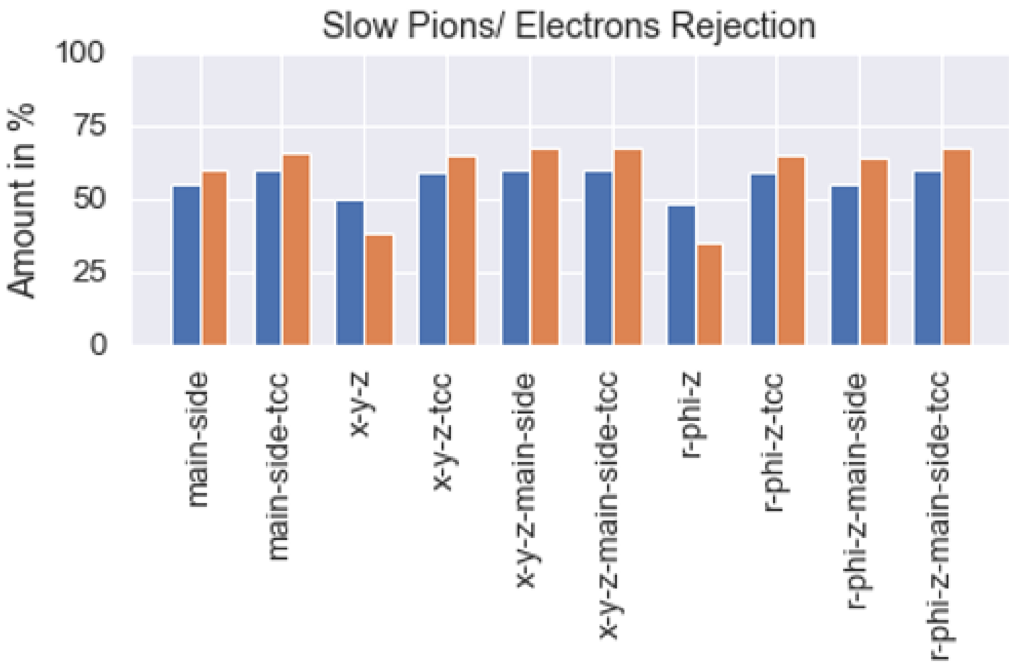
orange: rbf
blue: linear

Best: r-phi-z

Results: Rejection (Negative Predictive Value)

$$NPV = \frac{TN}{TN + FN}$$

"amount of electrons among particles classified as electrons"

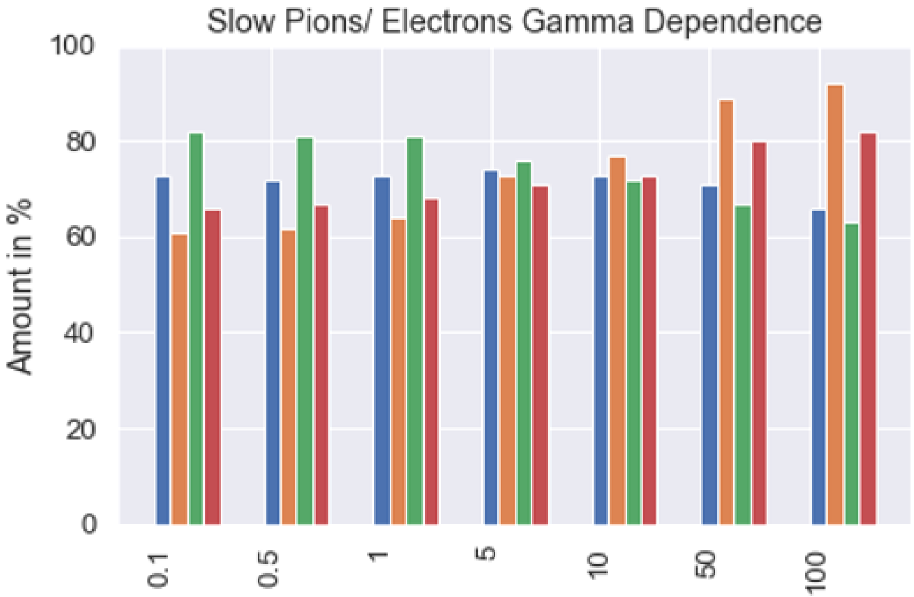


orange: rbf
blue: linear

Best: x-y-z-main-side-tcc

Gamma Dependence

Reminder: $$K(x,y) = exp(-\gamma \sum_{j=1}^{p}(x_{ij}-y_{ij})^2)$$



blue: accuracy
orange: purity
green: precision
red: rejection

Best: x-y-z-main-side-tcc

Summary

- Calculated values (accuracy, purity, efficiency, rejection) are not as good J. Bilk's ones
- SVMs show geometrical properties
- Further optimization possible by picking "the right" kernel

Thank you for your attention!