Clustering Methods for Excited State Promoted Readout

Mentee - Utkarsh

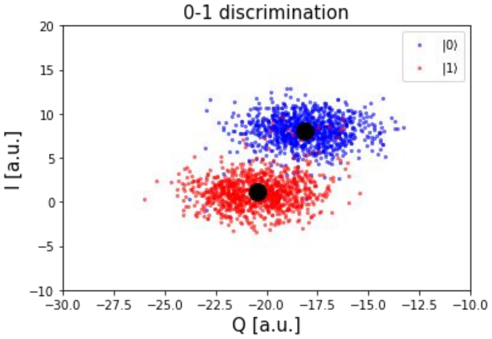
Mentor - Helena Zhang

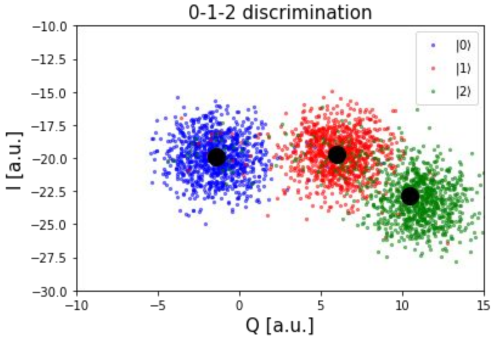
Project Description

1. Excited State Promoted (ESP) readout is a way to improve qubit readout fidelity by exciting the |1⟩ state to the |2⟩ state for readout. This essentially changes the discrimination problem from a two-state to three-state system.

2. In our project, we aim to look at different clustering algorithms on the real data to measure their effectiveness, and implement a method for effective discrimination.

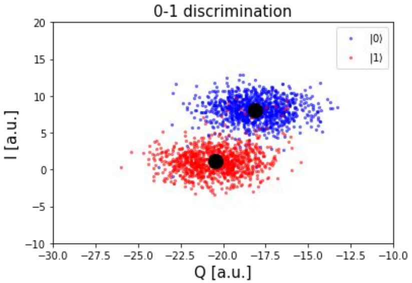
3. The metric for success is the false error rate on the labeled data.

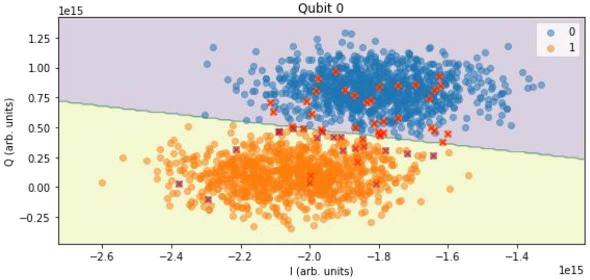


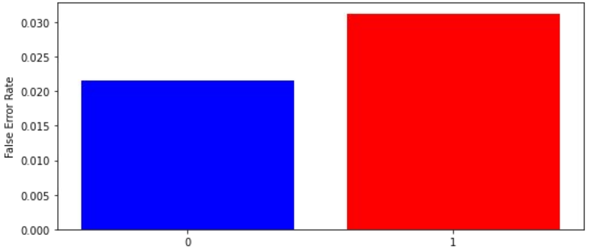


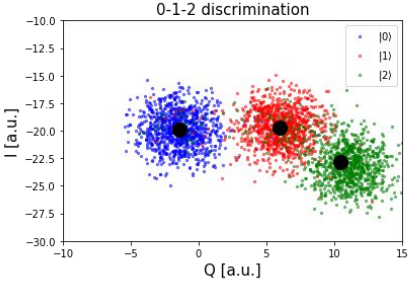
Current Progress - I

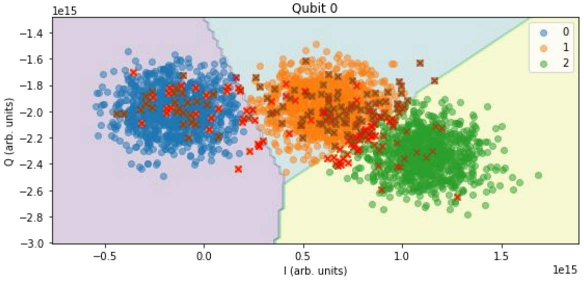
Got the basic Ignis discriminator setup working for the unexcited and excited state readout.

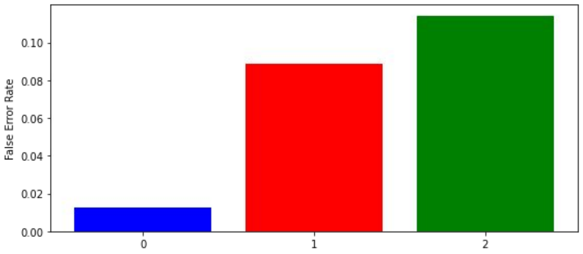








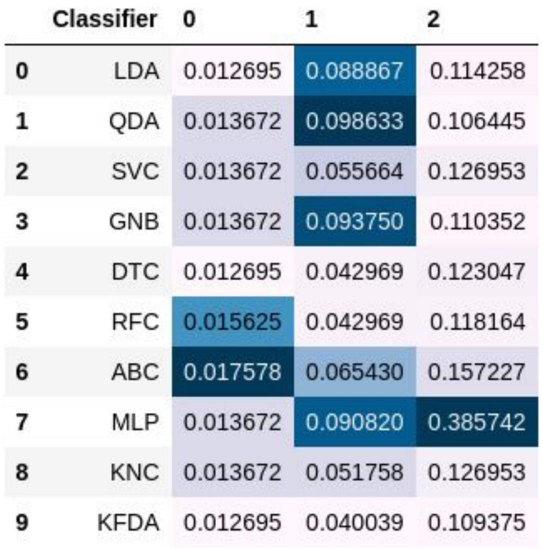




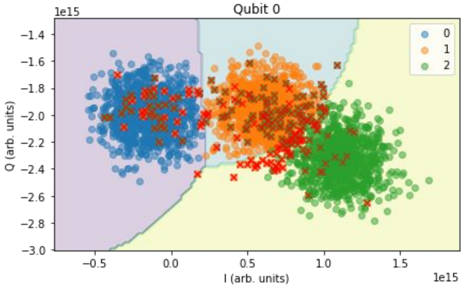
Current Progress - II

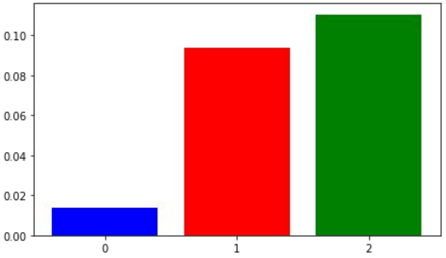
Comparison between the different *multiclass* classifiers.

1. Linear and Quadratic Discriminant Analysis
2. Support Vector Machines
3. Gaussian Naive Bayes
4. Decision Tree Classifiers and Random Forests
5. K-Nearest Neighbors
6. Multi-Layer Perceptrons and Ada Boost
7. Kernel Fisher Discriminant Analysis



	Classifier	0	1	2
0	LDA	0.012695	0.088867	0.114258
1	QDA	0.013672	0.098633	0.106445
2	SVC	0.013672	0.055664	0.126953
3	GNB	0.013672	0.093750	0.110352
4	DTC	0.012695	0.042969	0.123047
5	RFC	0.015625	0.042969	0.118164
6	ABC	0.017578	0.065430	0.157227
7	MLP	0.013672	0.090820	0.385742
8	KNC	0.013672	0.051758	0.126953
9	KFDA	0.012695	0.040039	0.109375



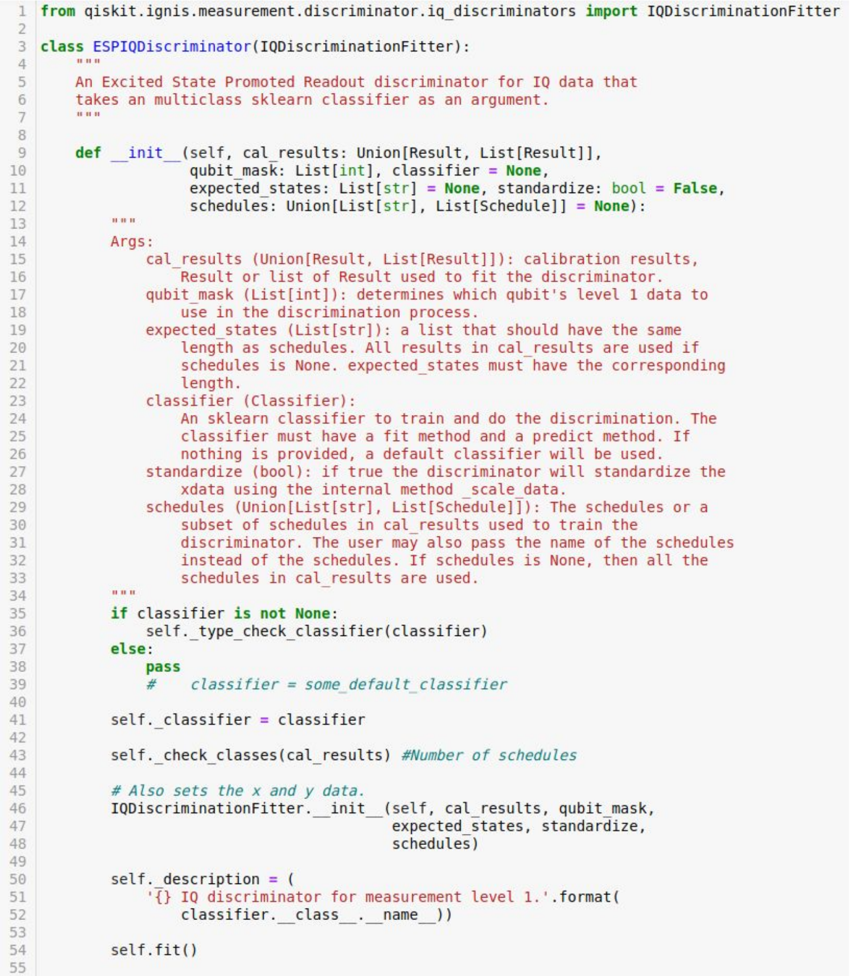


Current Progress - III

Attempting at an effective implementation of ESP discrimination.

Basic Idea -

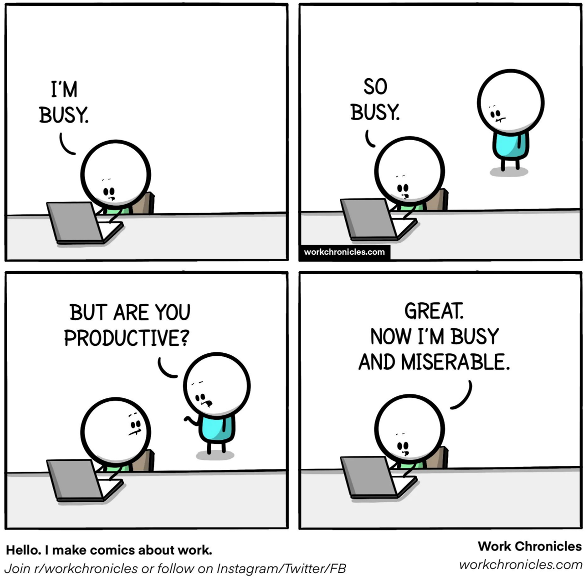
1. An **optional** multiclass classifier. If not provided by the user, choose an ideally best performing by **default**.
2. A check on **number of schedules** provided. Should be least *3*?
3. Should not break any of the existing functionalities in the discriminator code.

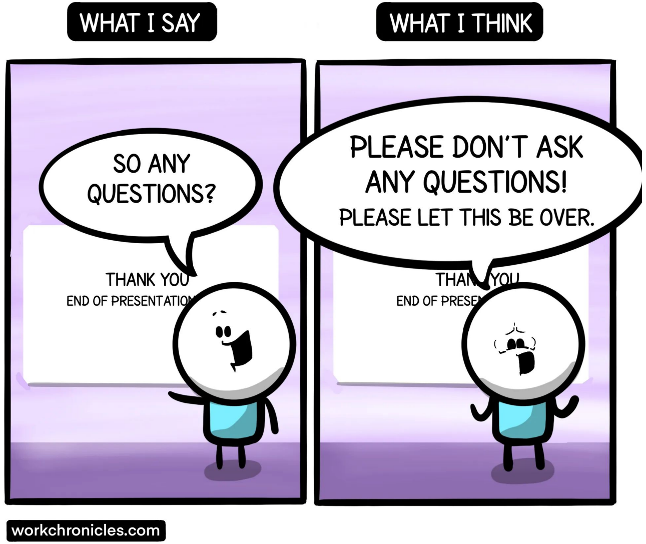


```python
from qiskit.ignis.measurement.discriminator.iq_discriminators import IQDiscriminationFitter

class ESPIQDiscriminator(IQDiscriminationFitter):
    """
    An Excited State Promoted Readout discriminator for IQ data that
    takes an multiclass sklearn classifier as an argument.
    """

    def __init__(self, cal_results: Union[Result, List[Result]],
                 qubit_mask: List[int], classifier = None,
                 expected_states: List[str] = None, standardize: bool = False,
                 schedules: Union[List[str], List[Schedule]] = None):
        """
        Args:
            cal_results (Union[Result, List[Result]]): calibration results,
                Result or list of Result used to fit the discriminator.
            qubit_mask (List[int]): determines which qubit's level 1 data to
                use in the discrimination process.
            expected_states (List[str]): a list that should have the same
                length as schedules. All results in cal_results are used if
                schedules is None. expected_states must have the corresponding
                length.
            classifier (Classifier):
                An sklearn classifier to train and do the discrimination. The
                classifier must have a fit method and a predict method. If
                nothing is provided, a default classifier will be used.
            standardize (bool): if true the discriminator will standardize the
                xdata using the internal method _scale_data.
            schedules (Union[List[str], List[Schedule]]): The schedules or a
                subset of schedules in cal_results used to train the
                discriminator. The user may also pass the name of the schedules
                instead of the schedules. If schedules is None, then all the
                schedules in cal_results are used.
        """
        if classifier is not None:
            self._type_check_classifier(classifier)
        else:
            pass
        #    classifier = some_default_classifier

        self._classifier = classifier

        self._check_classes(cal_results) #Number of schedules

        # Also sets the x and y data.
        IQDiscriminationFitter.__init__(self, cal_results, qubit_mask,
                                        expected_states, standardize,
                                        schedules)

        self._description = (
            '{} IQ discriminator for measurement level 1.'.format(
                classifier.__class__.__name__))

        self.fit()

```

Deliverables

1. A pull request for code that implements ESP discrimination.
2. Unit Test cases for the above implemented code.
3. Lots of hope for not breaking existing codebase.


I'M BUSY.
SO BUSY.
workchronicles.com
BUT ARE YOU PRODUCTIVE?
GREAT. NOW I'M BUSY AND MISERABLE.
Hello. I make comics about work.
Join r/workchronicles or follow on Instagram/Twitter/FB
Work Chronicles
workchronicles.com



Thank You.

Questions?