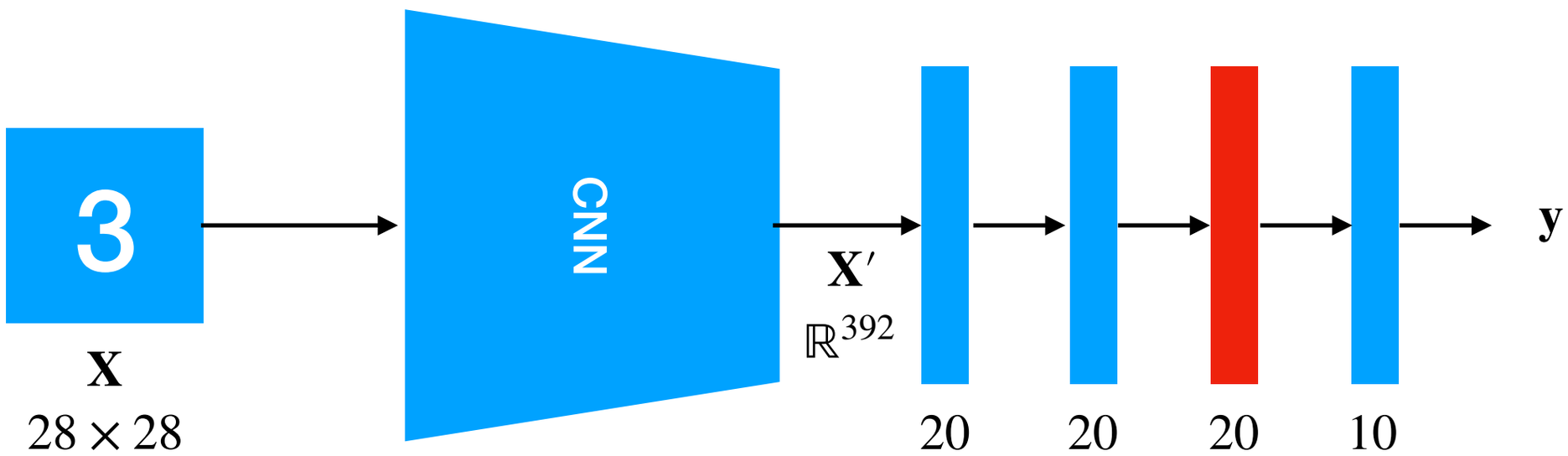
KDN Extended to CNNs (#4)
&
Weighted KDNs

KD-CNN



MNIST

KD-CNN

- Penultimate layer activations were used to determine the polytope memberships

- Instead of just neglecting singleton polytope, polytopes with members less than a specific threshold T (say, 10) were discarded (we could eliminate the thresholding using a weighting?)

- When looping across polytopes and fitting GMMs, only considered the unique polytopes

KD-CNN

```
for label in self.labels:
    print("label : ", label)
    self.polytope_means[label] = []
    self.polytope_cov[label] = []

    # Get all training items that match our given label
    X_ = X[np.where(y==label)[0]]

    # Calculate polytope memberships for each observation in X_
    polytopes = self._get_polytopes(X_)[0]
    unique_polytopes = np.unique(polytopes)
    print("Number of Polytopes : ", len(polytopes))
    print("Number of Unique Polytopes : ", len(np.unique(polytopes)))

    for polytope in unique_polytopes:
        # find all other data with same polytopes
        idx = np.where(polytopes==polytope)[0]

        if len(idx) < 10:
            continue  #skip all calculations if there are no other matching polytopes

        print("Number of Polytope members : ", len(idx))

        if self.criterion == None:
            # Calculate single Gaussian over data in group
            # Note: Will this break if we list 2+ covariance types?
            gm = GaussianMixture(n_components=1, covariance_type=self.covariance_types, reg_covar=1e-4).fit(X_[idx])
            self.polytope_means[label].append(
```

KD-CNN

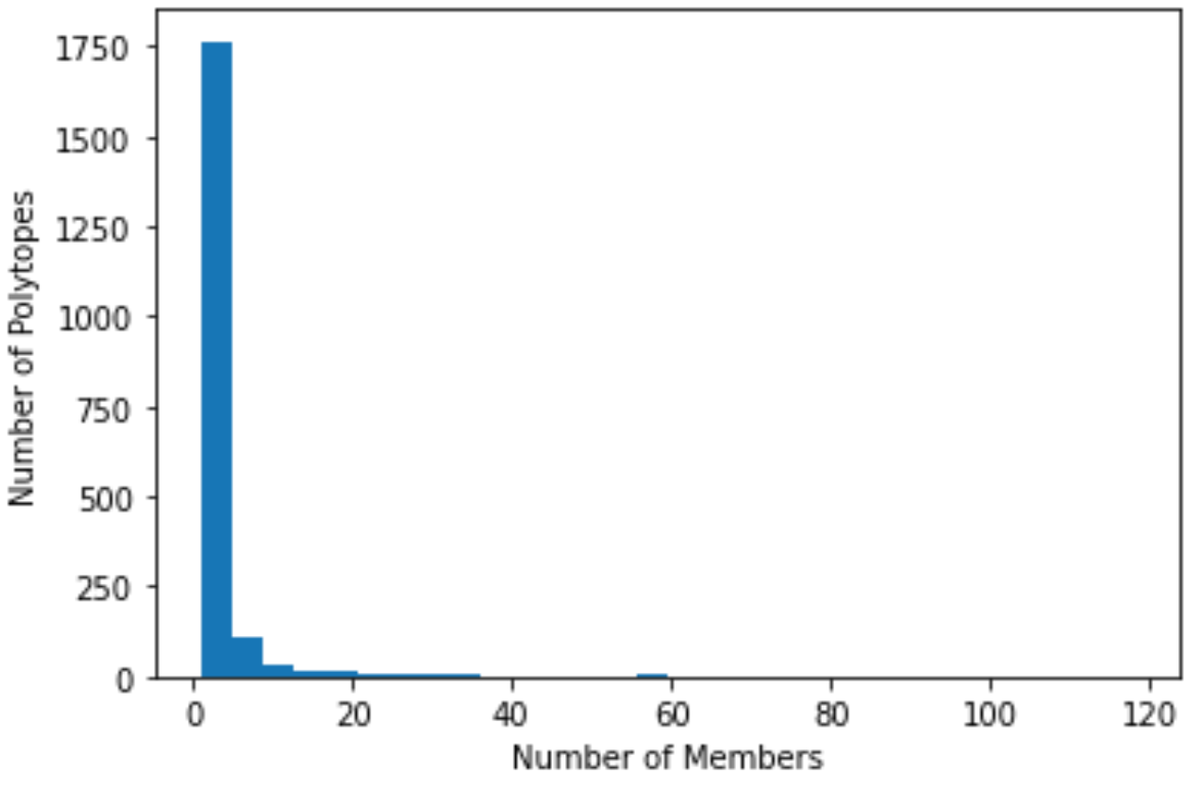
MNIST (latent dims = 392, 4 FC layers)

Model	Accuracy
Vanilla CNN	0.9671
KD-CNN (all the FC layers) $T = 1$	0.2574
KD-CNN (all the FC layers) $T = 10$	0.2574
KD-CNN (all the FC layers) $T = 100$	0.2574
KD-CNN (Penultimate FC layer) $T = 1$	0.9143
KD-CNN (Penultimate FC layer) $T = 10$	0.9143
KD-CNN (Penultimate FC layer) $T = 100$	0.9143

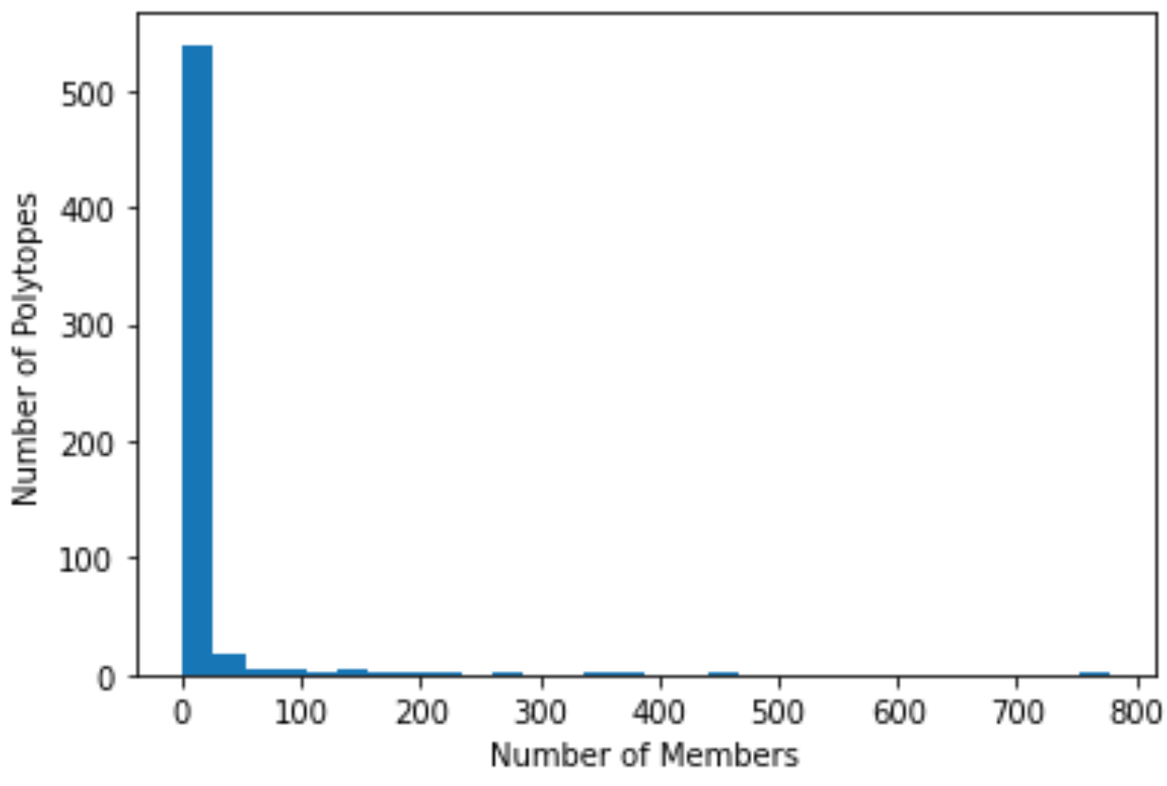
KD-CNN

When all the FC layers are used to compute polytope memberships

MNIST class 0



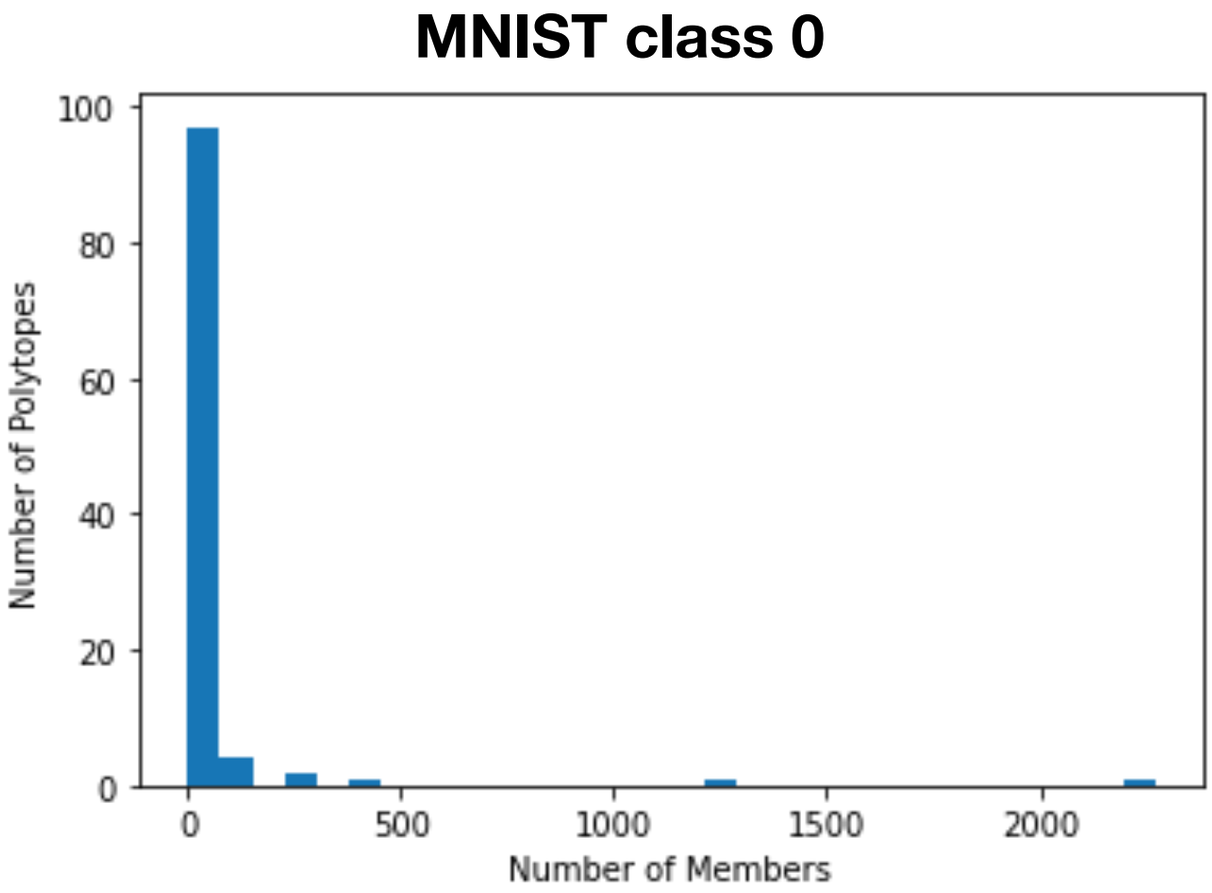
MNIST class 1

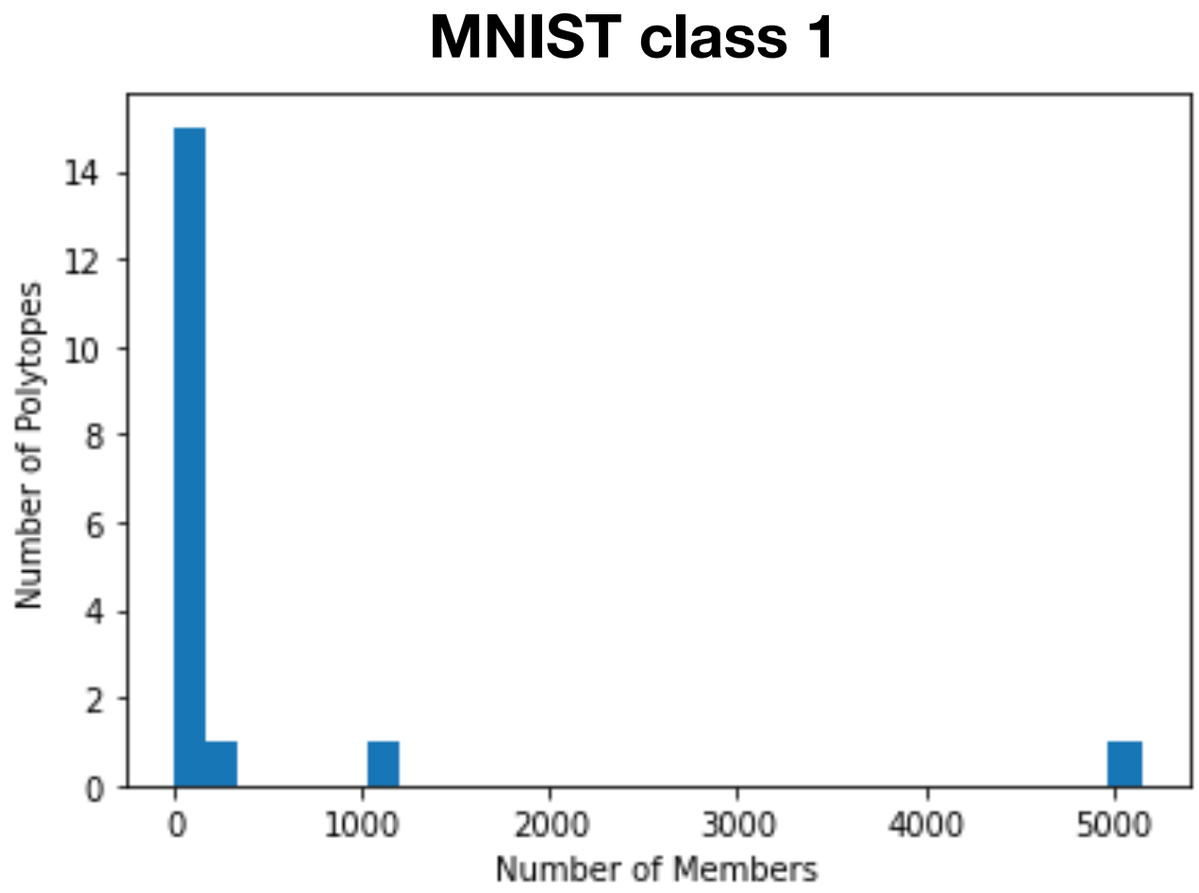


number of unique polytopes is high
members per polytope is generally low

KD-CNN

When only the penultimate layer is used to compute polytope memberships





number of unique polytopes is low
members per polytope is generally high

KD-CNN

CIFAR-10 (latent dims = 512)

Model	Accuracy
Vanilla CNN	0.6276
KD-CNN (all the FC layers) $T = 1$	-
KD-CNN (Penultimate FC layer) $T = 100$	0.4242

KD-CNN

CIFAR-10 (latent dims = 1024)

Model	Accuracy
Vanilla CNN	>>0.1
KD-CNN (all the FC layers) $T = 1$	-
KD-CNN (Penultimate FC layer) $T = 100$	0.1

KD-CNN

Issues

- High dimensional convolutional embeddings seems to problematic (high training/ inference times, low accuracy)

- How can we do the weighting of the covariance matrix in KDN/ KD-CNN?

- Penultimate layer vs. All the FC layers

KD-CNN

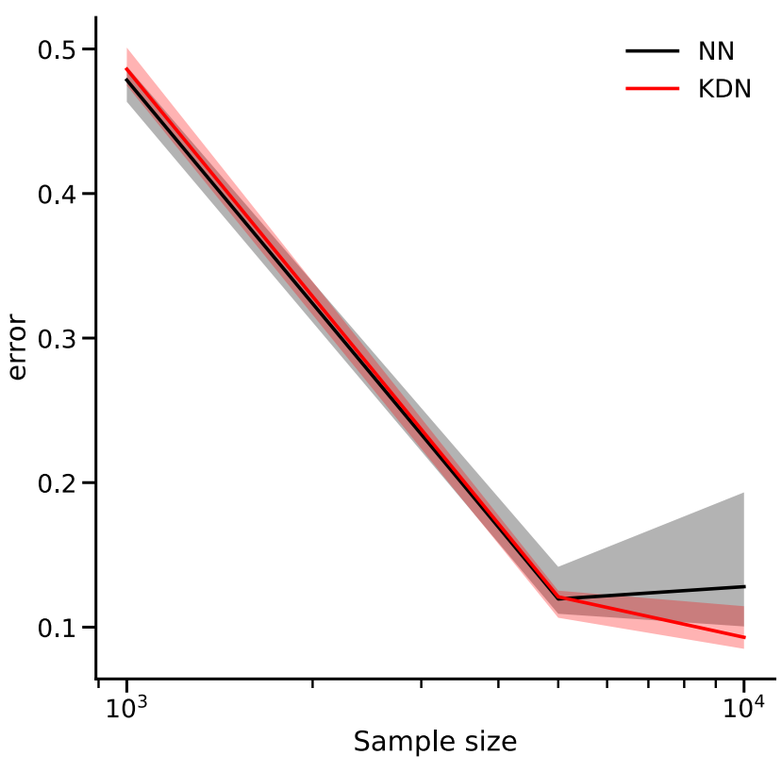
MNIST (latent dims = 392, 4 FC layers)

Model	Accuracy
Vanilla CNN	0.9671
KD-CNN (all the FC layers) $T = 10$	0.2574
KD-CNN (Penultimate FC layer) $T = 10$	0.9143
KD-CNN (all the FC layers + TM weighting)	0.9624
KD-CNN (all the FC layers + FM weighting)	0.9571

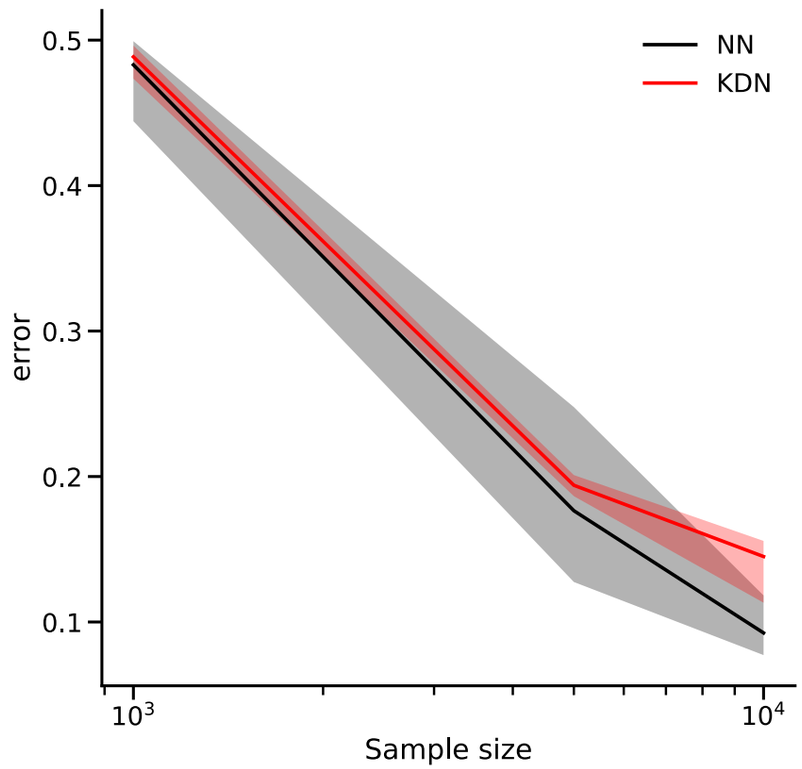
Weighted KDNs

Weighted KDNs

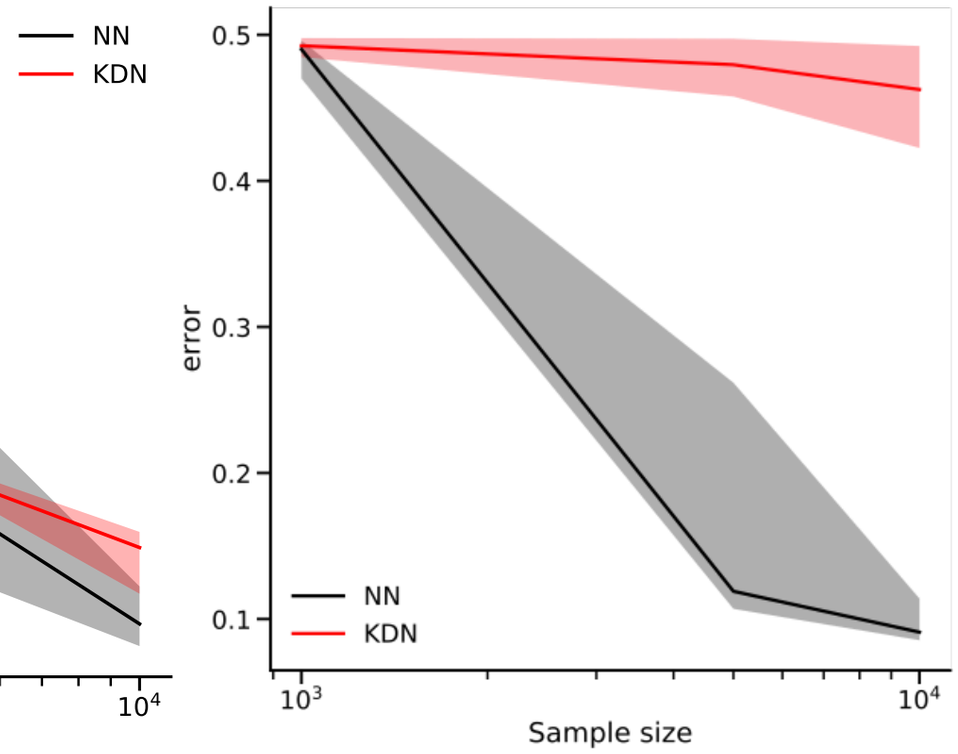
Evaluated on the Gaussian Sparse Parity Dataset (S3, N17)



KDN + allFC + No Weighting (5, 5, 2)



KDN + allFC + FM Weighting (5, 5, 2)

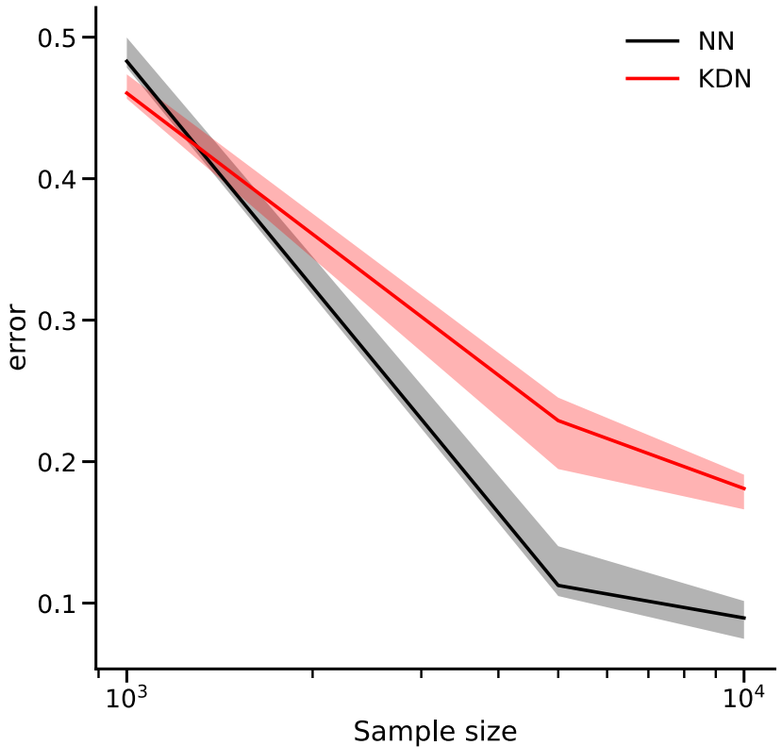


KDN + allFC + TM Weighting (5, 5, 2)

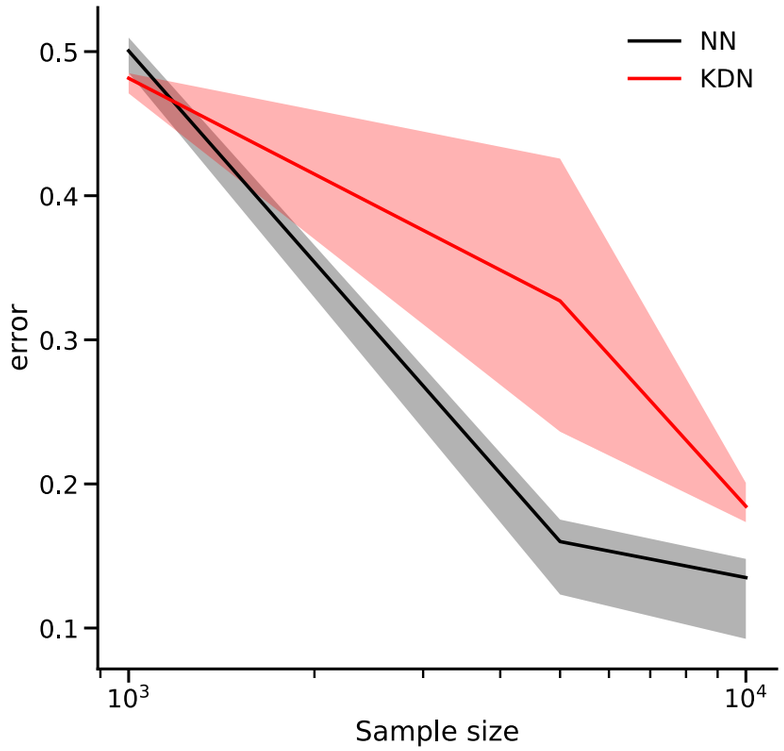
Same as LL-TM Weighting

Weighted KDNs

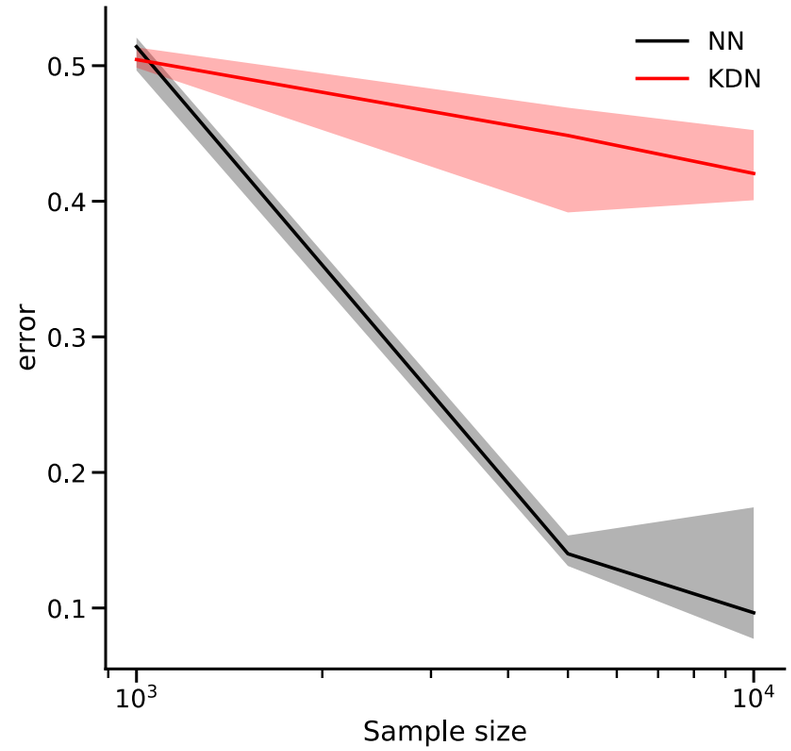
Evaluated on the Gaussian Sparse Parity Dataset (S3, N17)



KDN + PL + No Weighting (5, 5, 2)



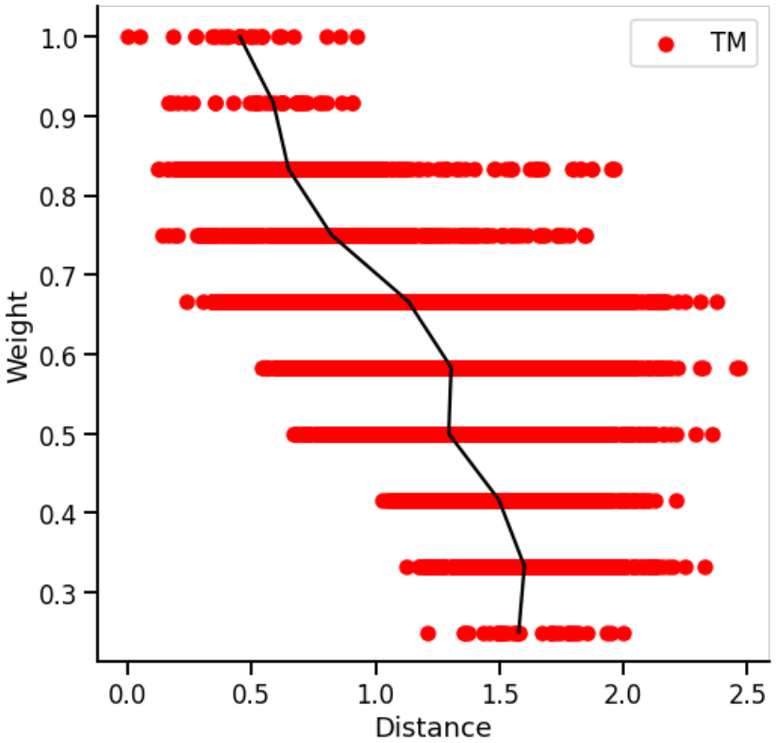
KDN + PL + FM Weighting (5, 5, 2)



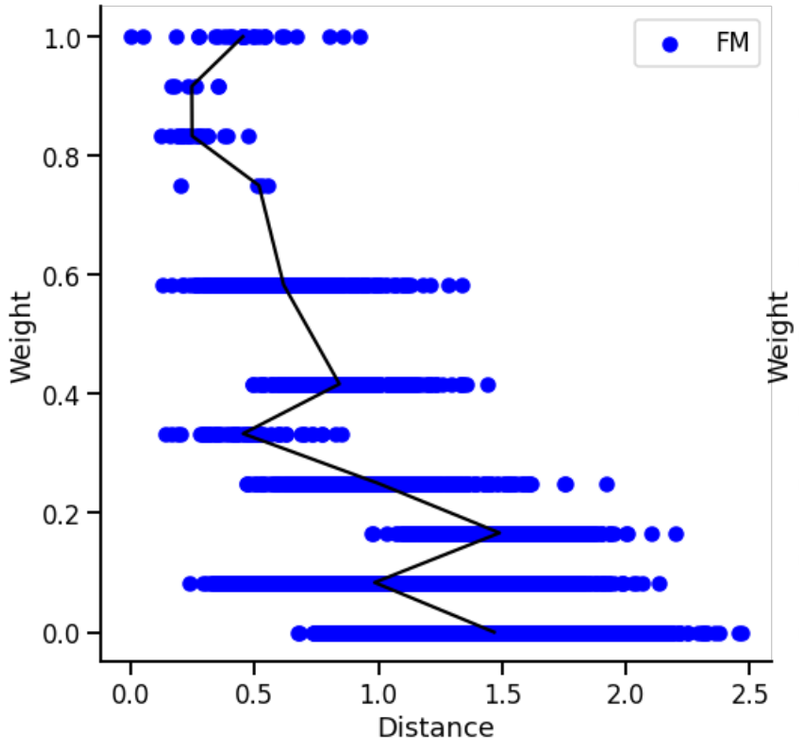
KDN + PL + TM Weighting (5, 5, 2)

Distance vs. Weight

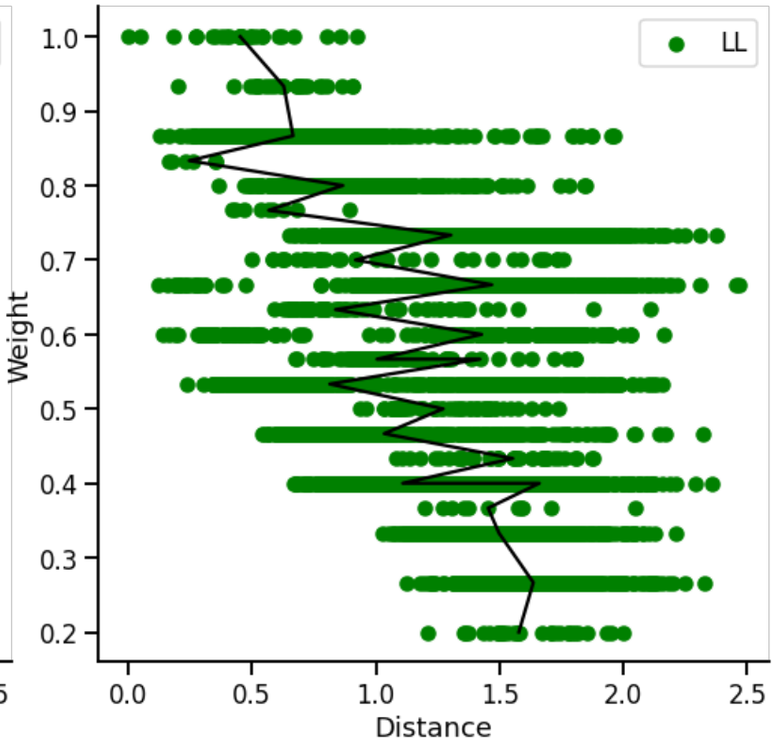
Evaluated on the Gaussian Sparse Parity Dataset (S3, N17)



TM Weighting
(5, 5, 2)
allFC



FM Weighting
(5, 5, 2)
allFC



LL Weighting
(5, 5, 2)
allFC