Conformal Regression with Reject Option

COPA 2024 - The 13th Symposium on Conformal and Probabilistic Prediction with Applications, Milano

Ulf Johansson, Cecilia Sönströd, Henrik Boström

11.09.2024

Jönköping University and KTH Royal Institute of Technology, Sweden

JÖNKÖPING UNIVERSITY

School of Engineering

Why reject option?

The *prediction with reject option* framework introduces the option for a model to refrain from making predictions for instances where uncertainty is high

In a human-in-the-loop scenario:

- predictive model is used for easy instances; difficult instances are referred to a human expert
- essential to accurately determine which instances to predict using the model
- desirable to know the human workload in advance

Underlying assumption: error on predictions made should decrease as the proportion of rejected instances increases

So, predictive quality can be increased by decreasing coverage, i.e. the proportion of instances for which predictions are delivered

The prediction with reject option framework thus offers a methodological framework for formalizing the trade-off between predictive performance and coverage

Allows design of human-in-the-loop systems with shared workload

Regression with reject option

Rejecting a prediction is based of the estimated uncertainty of the prediction

Defined by an uncertainty function $g(\boldsymbol{x_i})$ and a threshold τ

Formally, a regressor with reject option includes an additional output:

$$m(\boldsymbol{x_i}) = \begin{cases} ® & \text{if } g(\boldsymbol{x_i}) \geq \tau \\ h(\boldsymbol{x_i}) & \text{if } g(\boldsymbol{x_i}) < \tau. \end{cases} \tag{1}$$

where $m(\boldsymbol{x_i})$ is the output from the regressor with reject option and $h(\boldsymbol{x_i})$ is the prediction from the underlying model

The threshold τ is used to control the trade-off between error, e.g. measured as MAE, and prediction coverage.

Our suggested approach equips a standard conformal regressor with a reject option - meaning that the output from predicted instances are prediction intervals

In contrast, Sokol et al[1], studied regression with reject option, using conformalized quantile regression with interval size as the difficulty estimation of an instance

In Sokol et al, the conformal prediction was used as a tool for rejecting instances with high uncertainty

[1] A. Sokol, N. Moniz, and N. Chawla, *Conformalized selective regression*, arXiv 2402.16300, 2024

Mondrian Conformal Regression

Mondrian Conformal Regression: Instances are partitioned into categories and then ICP is applied to each category separately.

Every category requires its own calibration set, but also that the validity guarantees apply to each category independently.

Even without normalization, Mondrian approach makes the conformal regressor sharper since interval sizes differ between categories

Main contribution

1. A test prediction should be either a prediction interval or Ⓡ (reject).
2. The non-rejected predictions should be valid in the standard conformal sense, i.e., the error rate of the predicted intervals should be ϵ.
3. The efficiency of the non-rejected intervals should increase, i.e., the intervals should be tighter, when the regressor is allowed to reject more instances.
4. It should be possible for a user to know the interval sizes for any combination of significance and rejection levels after the calibration step, i.e., before making the first test prediction.

Key contribution: employ Mondrian conformal regression where the categories are determined from a difficulty estimator

More technically: For a chosen rejection level ρ, e.g., 0.1, the Mondrian taxonomy dictates that all test instances with a higher difficulty than the calibration instance corresponding to the $(1-\rho)$-percentile difficulty estimate in the calibration set should belong to Category ® and the remaining to Category P.

Test instances belonging to Category ® are rejected, while the ones belonging to Category P are predicted.

Prediction intervals for the predicted instances are generated using standard ICP, i.e., without normalization.

Experimental setup

Regressors: random forests as implemented in scikit learn, with 300 trees

Conformal regressors: Crepes package, with out-of-bag calibration

Difficulty estimators: four options, all available in Crepes

- **kNN distance**: average distance to 25 nearest neighbors - less populated parts of feature space should be harder
- **kNN variance**: standard deviation of target values of 25 nearest neighbors - instances where neighboring target values vary more should be harder
- **kNN residuals**: mean of the absolute residuals of the 25 nearest neighbors - instances where the model is less accurate on neighbors are expected to be harder
- **Tree variance**: variance of the individual tree predictions - instances are estimated to be harder the more the trees disagree

Testing protocol: standard 10-fold cross-validation.

Data sets: 16 publicly available medium-sized data sets ranging from approximately 4200 to 9500 instances, all targets scaled to $[0, 1]$.

Name	#inst.	#attrib.	Origin	Name	#inst.	#attrib.	Origin
abalone	4177	8	UCI	kin8fh	8192	8	Delve
bank8fh	8192	8	Delve	kin8fm	8192	8	Delve
bank8fm	8192	8	Delve	kin8nh	8192	8	Delve
bank8nh	8192	8	Delve	kin8nm	8192	8	Delve
bank8nm	8192	8	Delve	puma8fh	8192	8	Delve
comp	8192	12	Delve	puma8fm	8192	8	Delve
deltaA	7129	5	KEEL	puma8nh	8192	8	Delve
deltaE	9517	6	KEEL	puma8nm	8192	8	Delve

kin data sets

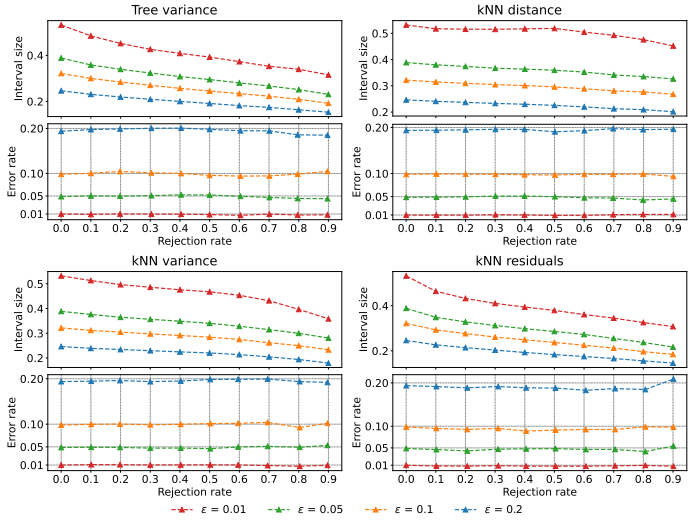
Four *kin* data sets, all variations on the same model; a realistic simulation of the forward dynamics of an eight link all-revolute robot arm.

The task is to predict the distance of the end-effector from a target, where inputs include joint positions, twist angles, etc.

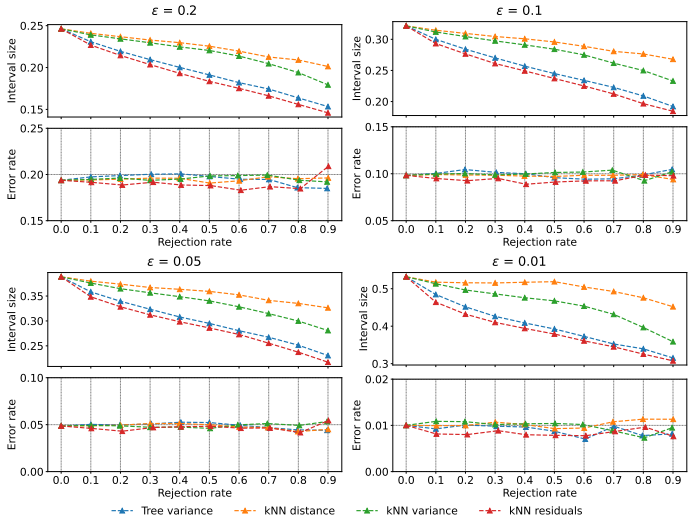
The four data sets differ in two ways:

1. they are either "fairly linear" (identified by the letter *f* in the data set name) or "non-linear" (letter *n* in data set name)
2. the noise level is either "medium unpredictability/noise" (letter *m* in data set name) or "high unpredictability/noise" (letter *h* in data set name).

Results for kin8nm data set - overview



Results for kin8nm data set - estimator comparison



Empirical rejection levels

Result: empirical rejection levels are, for all four difficulty estimation functions, almost identical to the requested

	Rejection level								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
Tree variance	.100	.201	.300	.400	.499	.600	.700	.800	.900
kNN distance	.102	.202	.303	.401	.501	.602	.701	.801	.900
kNN variance	.100	.199	.300	.400	.500	.600	.700	.800	.900
kNN residuals	.100	.199	.300	.400	.499	.600	.700	.800	.899

Table 1: Empirical rejection rates per rejection level - Over all data sets

Expectation: point prediction errors should decrease with increasing reject proportion

	Rejection level									
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Tree variance	.057	.054	.052	.049	.047	.044	.042	.039	.036	.033
kNN distance	.057	.056	.055	.054	.053	.052	.052	.051	.050	.049
kNN variance	.057	.055	.052	.051	.049	.047	.045	.044	.041	.038
kNN residuals	.057	.054	.052	.050	.048	.047	.045	.043	.041	.038

Table 2: Mean absolute errors per rejection level

Empirical error rates, $\epsilon = 0.2, 0.1, 0.05$

Result: empirical error rates very close to the significance levels.

Overall tendency to be slightly conservative, as expected when using out-of-bag calibration.

		Rejection level									
		0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
$\epsilon = 0.2$	Tree variance	.199	.199	.200	.198	.198	.199	.198	.200	.200	.203
	kNN distance	.199	.199	.199	.199	.198	.198	.197	.198	.199	.198
	kNN variance	.199	.199	.198	.198	.198	.197	.196	.199	.198	.201
	kNN residuals	.199	.198	.198	.197	.196	.197	.197	.198	.203	.204
$\epsilon = 0.1$	Tree variance	.099	.099	.100	.099	.099	.099	.098	.098	.100	.100
	kNN distance	.099	.099	.099	.099	.099	.100	.098	.099	.098	.096
	kNN variance	.099	.099	.099	.099	.099	.098	.098	.100	.100	.100
	kNN residuals	.099	.098	.098	.099	.098	.099	.099	.100	.101	.100
$\epsilon = 0.05$	Tree variance	.049	.049	.050	.049	.049	.049	.049	.049	.050	.051
	kNN distance	.049	.049	.049	.049	.049	.049	.049	.050	.049	.050
	kNN variance	.049	.049	.049	.050	.049	.048	.048	.050	.050	.051
	kNN residuals	.049	.049	.049	.050	.049	.049	.049	.050	.050	.050

Interval sizes, predicted instances. $\epsilon = 0.2, 0.1, 0.05$

A key component of the suggested method is the possibility for a user to know the interval sizes of future test set predictions for different significance and rejection levels.

		Rejection level									
		0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
$\epsilon = 0.2$	Tree variance	.178	.167	.158	.150	.142	.134	.125	.116	.106	.095
	kNN distance	.178	.173	.170	.167	.164	.162	.159	.156	.153	.150
	kNN variance	.178	.169	.162	.156	.150	.144	.138	.131	.124	.113
	kNN residuals	.178	.168	.161	.155	.149	.143	.137	.131	.123	.112
$\epsilon = 0.1$	Tree variance	.238	.223	.211	.200	.189	.179	.168	.157	.144	.131
	kNN distance	.238	.231	.227	.222	.218	.214	.211	.207	.203	.197
	kNN variance	.238	.225	.215	.207	.199	.192	.185	.176	.166	.153
	kNN residuals	.238	.224	.214	.206	.198	.190	.183	.174	.164	.152
$\epsilon = 0.05$	Tree variance	.292	.274	.260	.248	.236	.224	.211	.198	.184	.169
	kNN distance	.292	.284	.278	.273	.268	.264	.259	.254	.249	.243
	kNN variance	.292	.276	.264	.254	.245	.237	.229	.219	.207	.191
	kNN residuals	.292	.274	.262	.252	.243	.234	.226	.215	.204	.192

Correlations between difficulty estimates and errors

	Tree variance	kNN distance	kNN variance	kNN residuals
abalone	.380	.214	.350	.365
bank8fh	.279	.122	.188	.141
bank8fm	.461	.303	.360	.320
bank8nh	.264	.209	.181	.183
bank8nm	.599	.478	.553	.574
comp	.399	.274	.326	.433
deltaA	.413	.324	.409	.425
deltaE	.193	.108	.166	.155
kin8fh	.177	.035	.192	.194
kin8fm	.206	.180	.248	.401
kin8nh	.207	.058	.198	.235
kin8nm	.333	.115	.165	.431
puma8fh	.261	-.035	.266	.249
puma8fm	.301	-.045	.297	.276
puma8nh	.346	.072	.262	.228
puma8nm	.322	.040	.281	.213
Mean	**.321**	**.153**	**.278**	**.301**

Results summary

Summarizing the aggregated results, it is seen that also over all data sets the method works well

Empirical error and rejection rates match requested levels

Interval sizes decrease overall, as rejection level is increased, empirically demonstrating that this desired efficiency property holds

However, rather weak correlation between test set difficulty estimations and test set errors

Concluding remarks

Introduced and evaluated conformal regression with reject option

Key ideas are that a test prediction should either be a prediction interval or a reject and that the non-rejected predictions are valid.

Empirical evaluation demonstrated these properties, and also that all difficulty estimators produced tighter intervals the more instances the regressor was allowed to reject.

From a practitioner's perspective, the suggested setup makes it possible to compare the interval sizes resulting from combinations of significance and rejection levels before making any predictions.

Concluding remarks

Four different difficulty estimators; three based on neighboring instances, and one on disagreement between the trees in the Random forest

All difficulty estimators worked as intended, i.e., produced orderings that were good enough to make the intervals tighter for higher rejection levels, correlation between difficulty estimates and prediction errors was not very high.

Obvious avenue for future work is exploring new and potentially stronger difficulty estimators.

Specifically, investigating different, and potentially varying, number of neighbors to consider would be a straightforward extension.

JÖNKÖPING UNIVERSITY