best4linreg.py

It is very clear that regression algorithm is more suitable for linear data.
Based on this idea, I create sets of data by adding linear coefficients to two features (X1,X2). Y is the combination of X1 and X2 and adding noise.

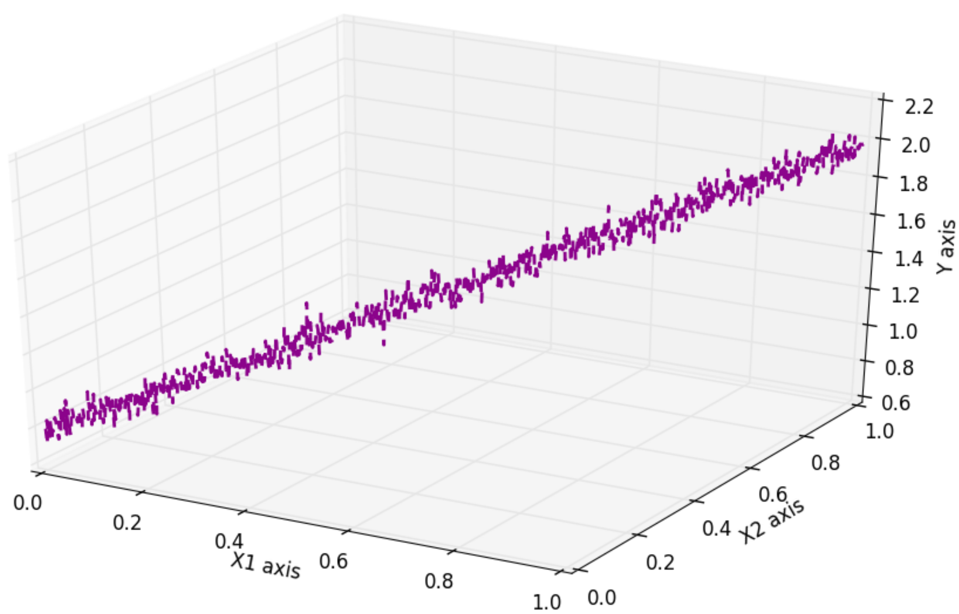
Summary:

First set of data:
Linear regression:
RMSE: 0.0494185434772
corr: 0.972078266397
Out of sample results
RMSE: 0.0495469537183
corr: 0.942602057829

KNN:
RMSE: 0.0395829330973
corr: 0.982177902937
Out of sample results
RMSE: 0.297606053586
corr: -1.4109006472e-15

Second set of data:
Linear regression:
In sample results
RMSE: 0.053251120801
corr: 0.97856152279
Out of sample results
RMSE: 0.0533116852068
corr: 0.959287677246

KNN:
In sample results
RMSE: 0.0424973064962
corr: 0.986392902156
Out of sample results
RMSE: 0.397603685459
corr: 1.2263851095e-16



Obviously, linear regression is much better than KNN(out of sample)

best4KNN.py

Linear regression algorithm is suitable for linear data. However, linear regression algorithm is bad for some non-linear data, especially classification data. KNN is better. And KNN works well with data with a high degree of similarity.

Based on these ideas, I create sets of classification data using certain condition setting.

Summary:

First set of data:

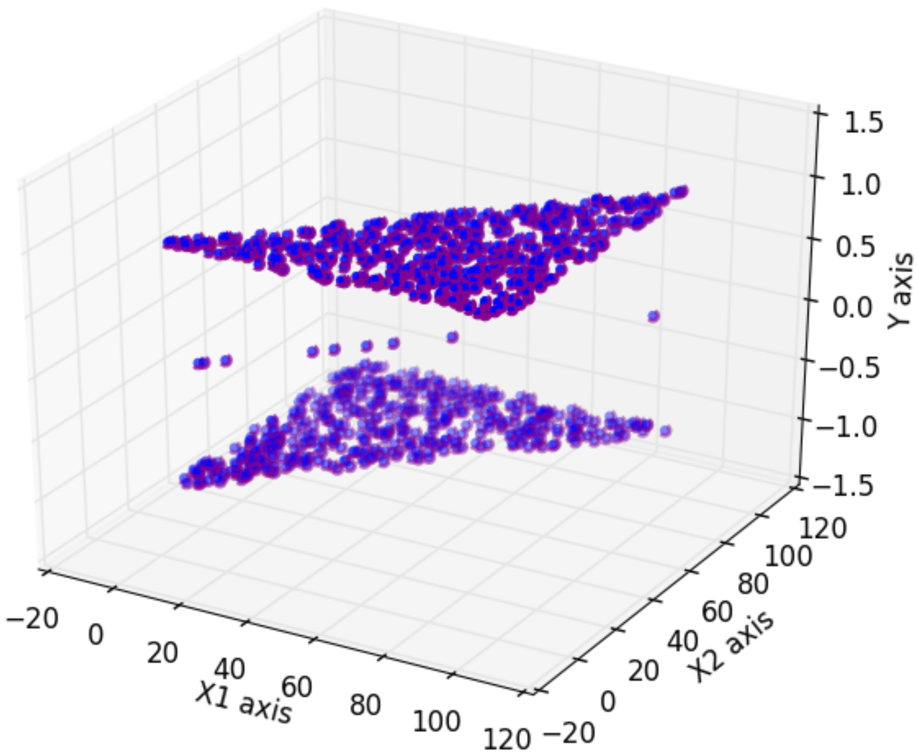
Linear regression:
In sample results
RMSE: 0.568147235283
corr: 0.820871575179
Out of sample results
RMSE: 0.588071491071
corr: 0.805957507265

KNN:
In sample results
RMSE: 0.117062819476
corr: 0.993054090797
Out of sample results
RMSE: 0.230940107676
corr: 0.972578731055

Second set of data:
Linear regression:
In sample results
RMSE: 0.560643758826
corr: 0.825962902372
Out of sample results
RMSE: 0.593680529881
corr: 0.80318686562

KNN:
In sample results
RMSE: 0.162731316843

corr: 0.986554847903
Out of sample results
RMSE: 0.266145323711
corr: 0.963733425897



Obviously, KNN is much better than linear regression.

Dataset ripple with KNN

When k=3:
RMSE: 0.136590187312
corr: 0.981360326901
Out of sample results
RMSE: 0.207762150054
corr: 0.955537498166

When k=2:

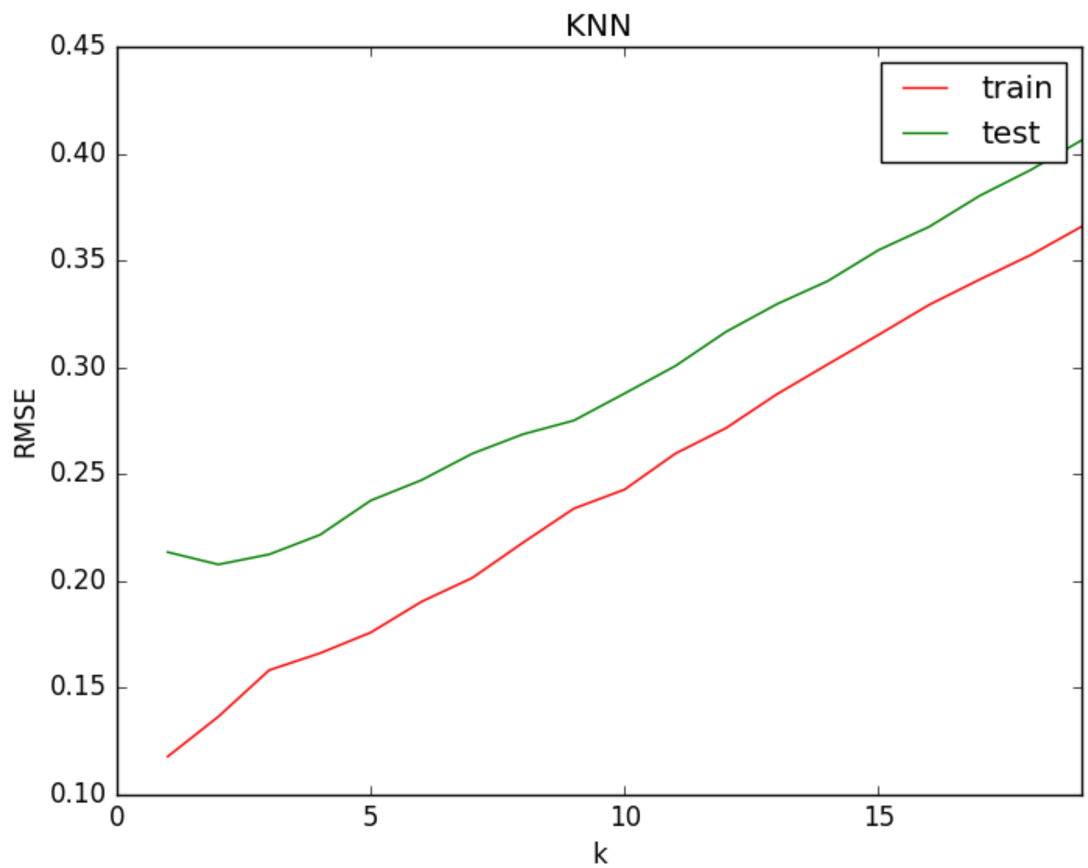
In sample results
RMSE: 0.117863434564
corr: 0.985971678512
Out of sample results
RMSE: 0.213547134947
corr: 0.952952269598

When k=1:
In sample results
RMSE: 0.0
corr: 1.0
Out of sample results
RMSE: 0.237716222234
corr: 0.944111176194

When k=4:
In sample results
RMSE: 0.158319703229
corr: 0.975383222588
Out of sample results
RMSE: 0.212486985302
corr: 0.954609910672

When k=5:
In sample results
RMSE: 0.166240346459
corr: 0.973427600347
Out of sample results
RMSE: 0.221620653532
corr: 0.951772965431

My plot:



Green line shows test RMS error
Red line shows train RMS error
When k=2, overfitting occurs because of error increasing again.

Dataset ripple with bagging using KNN (k=3)

When bags=10:
In sample results
RMSE: 0.131035007247
corr: 0.983972561687
Out of sample results
RMSE: 0.205467806723
corr: 0.957955547407

When bags=20:

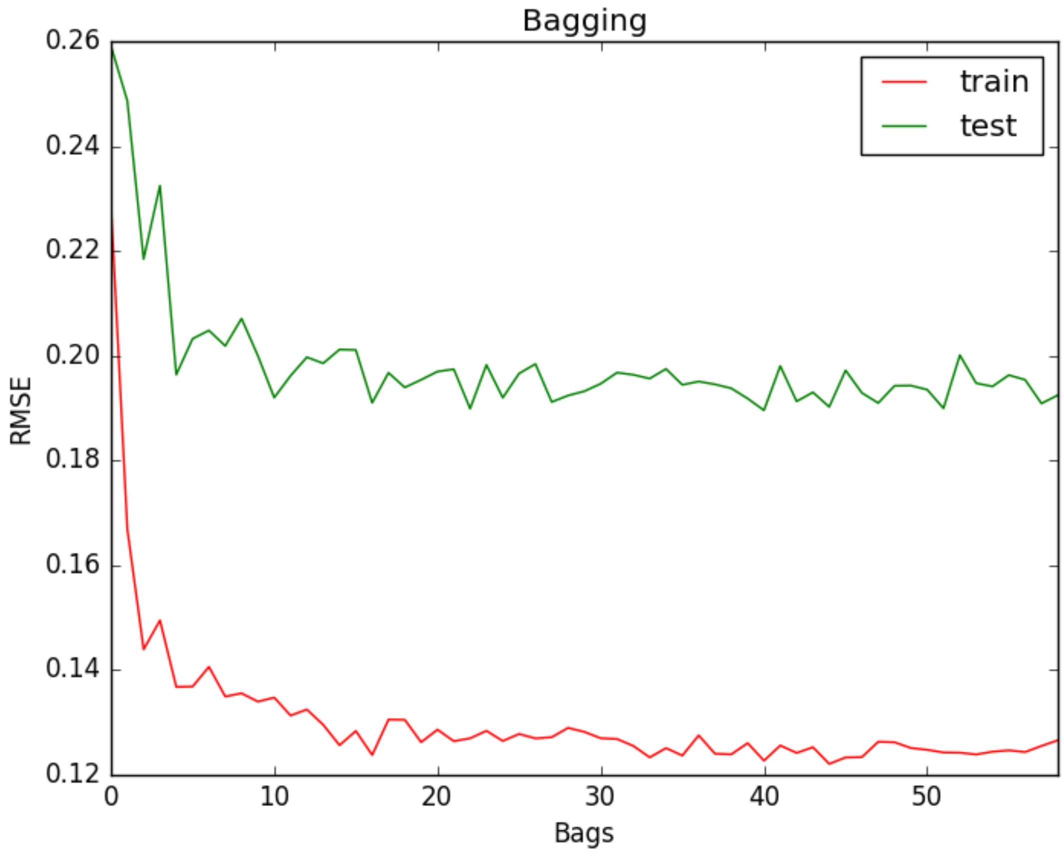
In sample results
RMSE: 0.12412424304
corr: 0.985949010275
Out of sample results
RMSE: 0.197590345316
corr: 0.961806775047

When bags=30:
In sample results
RMSE: 0.12701497184
corr: 0.985347638727
Out of sample results
RMSE: 0.196284214464
corr: 0.962706395727

When bags=40:
In sample results
RMSE: 0.125243144255
corr: 0.985697702653
Out of sample results
RMSE: 0.19426841615
corr: 0.963561311903

When bags=80:
In sample results
RMSE: 0.124449500729
corr: 0.985960721932
Out of sample results
RMSE: 0.191713861852

corr: 0.96477376994



Green line shows test RMS error
Red line shows train RMS error
When bag increases at the beginning, error decreases. When bag reaches 20, error keeps steady.
So no obvious overfitting occurs.

Reduce or eliminate overfitting with bagging?

When k=1 and bags=20:
In sample results
RMSE: 0.0781955782045
corr: 0.994162575957
Out of sample results
RMSE: 0.19163655086
corr: 0.962144251714

When k=2 and bags=20:
In sample results
RMSE: 0.10515924844
corr: 0.989530672926
Out of sample results
RMSE: 0.190765189734
corr: 0.963094723052

When k=3 and bags=20:
In sample results
RMSE: 0.129541123151
corr: 0.984423889466
Out of sample results
RMSE: 0.200474116165
corr: 0.961058714154

When k=4 and bags=20:
In sample results
RMSE: 0.144001311777
corr: 0.981696225157
Out of sample results
RMSE: 0.210077131545
corr: 0.957873903534

When k=5 and bags=20:
In sample results
RMSE: 0.164441111124
corr: 0.976688878882
Out of sample results
RMSE: 0.222765374697
corr: 0.955184515452
From these data, bagging can reduce overfitting with respect to K for the ripple dataset .