Solution of the PAKDDCup 2010

Michael Jahrer
[team : mjahrer]
commendo research & consulting
8580 Köflach
Austria
michael.jahrer@ commendo.at

ABSTRACT

We present the solution for team `mjahrer`. The ELF[4] project is used for training the prediction model. The final solution is a linear combination of two models, kernel ridge regression and gradient boosted decision trees. The given dataset is transformed to a 5840-feature numeric matrix by a simple encoding schema. The leaderboard AUC of our solution is 0.6249, the internal cross-validation score is 0.6609.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: [Data mining, Ensemble Learning]

Keywords

PAKDDCup, Supervised Learning, Ensemble Learning

1. INTRODUCTION

The data

The dataset stem from a Credit Risk Assessment System and has both numerical and categorical features. We have 50000 labeled training samples and two unlabeled sets (20000 for leaderboard feedback and 20000 for submission). Number of raw features is 53, they consist of numerical and categorical columns.

Transformation to numeric features

It is necessary to transform the categorical features into numeric inputs because the used learners can handle only dense numeric data. There are several features with constant content (e.g. CLERK_TYPE or EDUCATION_LEVEL) we skip them. And the first feature ID_CLIENT is also skipped. All numerical features f_i are added as $log(f_i + 1)$. For categorical features we build a histogram and take the tokens with an occurrence of larger equal 9 (simple one-hot encoding). This value was determined by many trials on a validation set. After the transformation of the 53 raw features we end up with 5864 numerical features. This implies that the training feature matrix has a size of 50000x5864. We do nothing special with the data (no feature selection, no data cleaning, etc.).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PAKDD2010 Hyderabad, INDIA
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$10.00.

Normalization of the features

We shift and scale every numeric feature to be in the [0...1] range.

The targets

This supervised problem has two different class labels, "BAD" or "GOOD". A good customer is able to pay his credit rates within a 60-day period. Target proportion is 26.082% (BAD) to 73.918% (GOOD).

Error function

Goal is to maximize the AUC on a test dataset. We optimize a linear combination of two algorithms in order to perform best on cross-validation. The linear combination was calculated by linear regression.

Model validation

Cross validation is used to do parameter selection of the learning models. We use 12-fold cross validation in our models. At the end of the parameter selection process, the model was re-trained on the whole training set in order to use all available data.

2. PREDICTION MODEL

We tried different models on the PAKDD2010 dataset. Our findings was that a k-nearest neighbor algorithm with euclidean distance measure does not work. Linear regression, kernel ridge regression, neural networks and gradient boosted decision trees gives good results. The most promising model is the kernel ridge regression algorithm because it shows the highest AUC score on the cross-validation set. At the end we combine two algorithms by using optimal linear combination.

Kernel Ridge Regression

The kernel ridge regression is a simple but computation expensive algorithm, because the matrix-inversion of the full gram matrix is costly (size NxN, where $N = 50000$). For the algorithm implementation see [3]. We use the gauss kernel (or RBF kernel) $k(\mathbf{a}, \mathbf{b}) = e^{\frac{||\mathbf{a}-\mathbf{b}||}{\sigma^2}}$. The kernel width is σ. As regularization we use ridge regression, controlled

by the parameter λ. The start values are $\sigma = 100$ and $\lambda = 1e-6$. We search for 50 epochs by using structured coordinate search.

Gradient Boosted Decision Tree

The algorithm is explained in detail by J. Friedman [1] [2]. Gradient boosting works epoch-wise, where only a fraction of the data is learned in an epoch. The learning rate for gradient boosting is η. Per split in the tree we consider K features (random subspace idea). Per epoch a tree is limited in the number of leafs, we allow S leafs in a tree. We train a minimum number of N_{epoch}, avoiding to stop the learning in the first local minima.

Combination

Two models are trained one after another. Firstly, the kernel ridge regression model is optimized in order to have the largest AUC score on the cross-validation set. Then we stop learning the gradient boosted decision tree when the linear combination of both models becomes maximal in terms of AUC. The linear combination is calculated with ridge regression.

3. RESULTS

model	AUC (blend)	RMSE (blend)	ACC (blend)	parameters, train time
KRR	0.6600	0.85123	25.97%	$\sigma = 10.1$, $\lambda = 4.4e-05$, 2.07[h]
GBDT	0.6609	0.85094	25.95%	$K = 100$, $S = 100$, $\eta = 0.005$, $N_{epoch} = 1500$, 72.4[h]

Table 1: Results of the two-model combination on the PAKDDCup2010 dataset. We report the AUC, RMSE and the accuracy (classification error) on a 12-fold cross-validation set.

The leaderboard score of our solution is 0.6249. It takes 1.8[h] to generate the leaderboard predictions. Blending weights of the final solution are: output $= 0.8 \cdot KRR + 0.23 \cdot GBDT$.

4. USED SOFTWARE

For the complete supervised learning process, including the data preprocessing, we used the ELF project [4]. The learning framework is freely available online, including a setup for reading, training and predicting the data from the PAKDDCup 2010.

5. REFERENCES

[1] J. Friedman. Greedy function approximation: A gradient boosting machine. Technical report, Salford Systems, 1999.

[2] J. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 2002.

[3] I. Guyon. *Kernel Ridge Regression tutorial*, accessed Aug 31, 2009. `http://clopinet.com/isabelle/Projects/ETH/KernelRidge.pdf`.

[4] M. Jahrer. ELF - Ensemble Learning Framework. An open source C++ framework for supervised learning. http://elf-project.sourceforge.net, 2010.

APPENDIX

Here are the configuration files for reproducing the results.

A. MASTER DSC FILE

```
dataset=PAKDDCup2010
isClassificationDataset=1

maxThreads=4
maxThreadsInCross=1
nCrossValidation=12
#validationType=CrossFoldMean
validationType=Retraining
#validationType=Bagging
positiveTarget=1.0
negativeTarget=-1.0
#negativeTarget=0.0
randomSeed=1234
nMixDataset=50
nMixTrainList=1000
standardDeviationMin=0.01
blendingRegularization=1e-5
blendingEnableCrossValidation=0
blendingAlgorithm=LinearRegression
enablePostNNBlending=0
enableCascadeLearning=0
enableGlobalMeanStdEstimate=0
enableSaveMemory=1
addOutputNoise=0
enablePostBlendClipping=0
enableFeatureSelection=0
featureSelectionWriteBinaryDataset=0
enableGlobalBlendingWeights=0
errorFunction=AUC
disableWriteDscFile=1
enableStaticNormalization=0
#staticMeanNormalization=0.0
#staticStdNormalization=1.0
#dimensionalityReduction=Autoencoder
enableProbablisticNormalization=1
addAutoencoderFeatures=0
subsampleTrainSet=1.0
subsampleFeatures=1.0
globalTrainingLoops=1
addConstantInput=1
gradientBoostingMaxEpochs=1
gradientBoostingLearnrate=1.0

[ALGORITHMS]
#LinearModel_1.dsc
#PolynomialRegression_1.dsc
#NeuralNetwork_1.dsc
#KNearestNeighbor_1.dsc
KernelRidgeRegression_1.dsc
GBDT_1.dsc
#NeuralNetwork_1.dsc
```

Table 2: Master configuration of the ELF.

B. KRR DSC FILE

```
ALGORITHM=KernelRidgeRegression
ID=1
#TRAIN_ON_FULLPREDICTOR=LinearModel_1.dat
DISABLE=0
enableProbablisticNormalization=1

[int]
maxTuninigEpochs=1

[double]
initMaxSwing=0.5
initReg=4.41051e-05

polyScaleInit=3.0
polyBiasPosInit=0.5
polyBiasNegInit=0.1
polyPowerInit=4.0

gaussSigmaInit=10.1148

tanhScaleInit=0.5
tanhBiasPosInit=0.5
tanhBiasNegInit=0.5

[bool]
enableClipping=0
enableTuneSwing=0

minimzeProbe=0
minimzeProbeClassificationError=0
minimzeBlend=1
minimzeBlendClassificationError=0

[string]
kernelType=Gauss
weightFile=KernelRidgeRegression_1_weights.dat
fullPrediction=KernelRidgeRegression_1.dat
```

Table 3: KRR configuration.

C. GBDT DSC FILE

```
ALGORITHM=GBDT
ID=2
#TRAIN_ON_FULLPREDICTOR=LinearModel_1.dat
DISABLE=0

[int]
minTuninigEpochs=1500
maxTuninigEpochs=2000

featureSubspaceSize=100
maxTreeLeafes=100

[double]
initMaxSwing=1.0
lRate=0.005

[bool]
calculateGlobalMean=1
useOptSplitPoint=1
enableClipping=0
enableTuneSwing=0

# minimize probe/blend RMSE
minimzeProbe=0
minimzeProbeClassificationError=0
minimzeBlend=1
minimzeBlendClassificationError=0

[string]
weightFile=GBDT_1_weights.dat
fullPrediction=GBDT_1.dat
```

Table 4: GBDT configuration.