Combining predictors

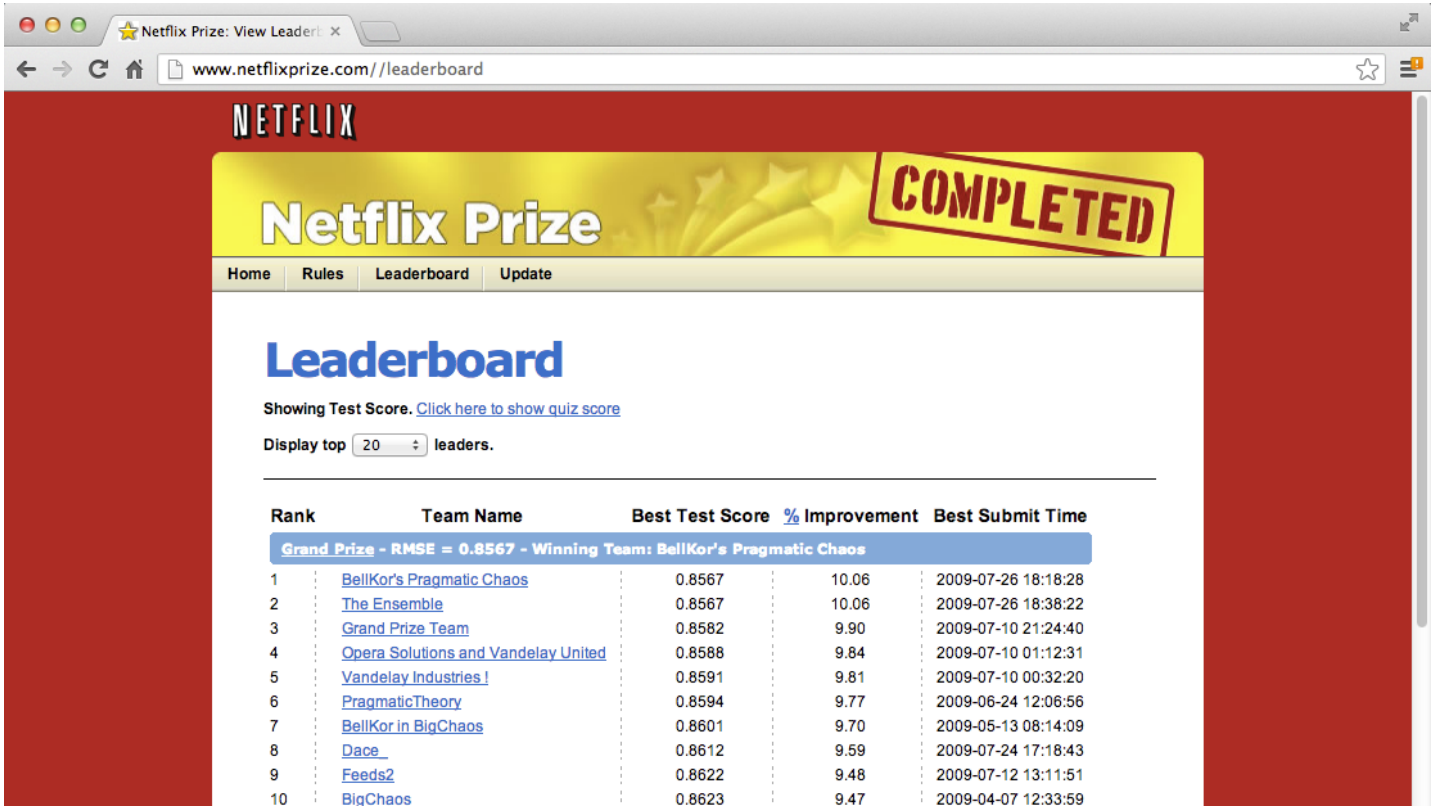
Jeffrey Leek, Assistant Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Key ideas

- You can combine classifiers by averaging/voting
- Combining classifiers improves accuracy
- Combining classifiers reduces interpretability
- Boosting, bagging, and random forests are variants on this theme

Netflix prize

BellKor = Combination of 107 predictors



http://www.netflixprize.com//leaderboard

Heritage health prize - Progress Prize 1

2. *Predictive Modelling*
   Predictive models were built utilising the data sets created in Step 1. Numerous mathematical techniques were used to generate a set of candidate solutions.

3. *Ensembling*
   The individual solutions produced in Step 2 were combined to create a single solution that was more accurate than any of its components.

Market Makers

1 Introduction

My milestone 1 solution to the Heritage Health Prize with a RMSLE score of 0.457239 on the leaderboard consists of a linear blend of 21 result. These are mostly generated by relatively simple models which are all trained using stochastic gradient descent. First in section 2 I provide a description of the way the data is organized and the features that were used. Then in section 3 the training method and the post-processing steps are described. In section 4 each individual model is briefly described, all the relevant meta-parameter settings can be found in appendix Parameter settings. Finally the weights in the final blend are given in section 5.

Mestrom

Basic intuition - majority vote

Suppose we have 5 completely independent classifiers

If accuracy is 70% for each:

- $10 \times (0.7)^3(0.3)^2 + 5 \times (0.7)^4(0.3)^2 + (0.7)^5$
- 83.7% majority vote accuracy

With 101 independent classifiers

- 99.9% majority vote accuracy

Approaches for combining classifiers

1. Bagging, boosting, random forests
   - Usually combine similar classifiers
2. Combining different classifiers
   - Model stacking
   - Model ensembling

Example with Wage data

Create training, test and validation sets

```
library(ISLR); data(Wage); library(ggplot2); library(caret);
Wage <- subset(Wage,select=-c(logwage))

# Create a building data set and validation set
inBuild <- createDataPartition(y=Wage$wage,
                               p=0.7, list=FALSE)
validation <- Wage[-inBuild,]; buildData <- Wage[inBuild,]

inTrain <- createDataPartition(y=buildData$wage,
                               p=0.7, list=FALSE)
training <- buildData[inTrain,]; testing <- buildData[-inTrain,]
```

Wage data sets

Create training, test and validation sets

```
dim(training)
```

```
[1] 1474   11
```

```
dim(testing)
```

```
[1] 628  11
```

```
dim(validation)
```

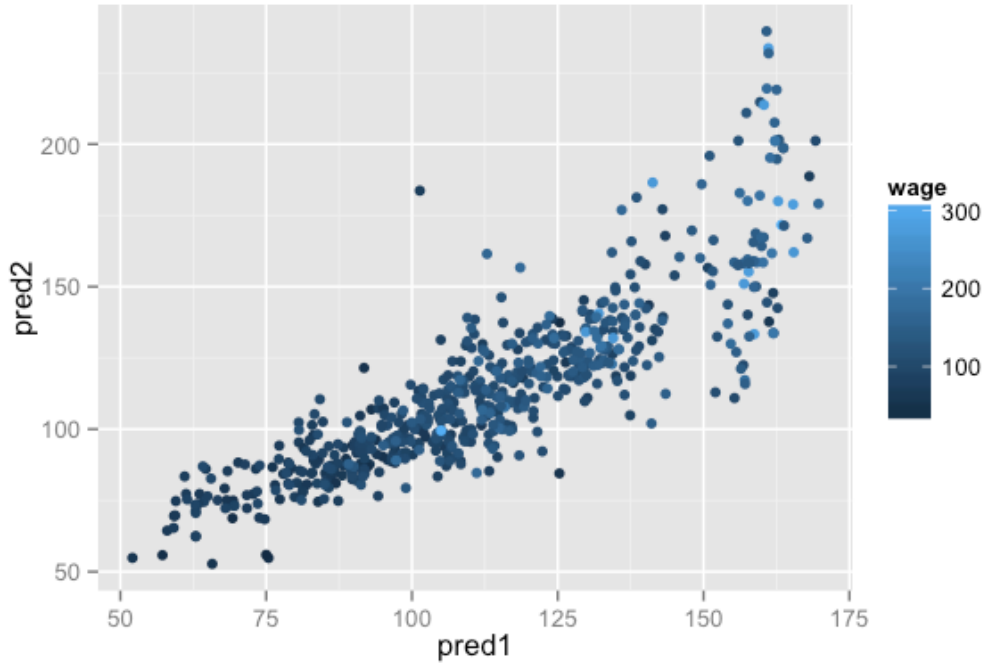
```
[1] 898  11
```

Build two different models

```
mod1 <- train(wage ~.,method="glm",data=training)
mod2 <- train(wage ~.,method="rf",
              data=training,
              trControl = trainControl(method="cv"),number=3)
```

Predict on the testing set

```
pred1 <- predict(mod1,testing); pred2 <- predict(mod2,testing)
qplot(pred1,pred2,colour=wage,data=testing)
```



Fit a model that combines predictors

```
predDF <- data.frame(pred1,pred2,wage=testing$wage)
combModFit <- train(wage ~.,method="gam",data=predDF)
combPred <- predict(combModFit,predDF)
```

Testing errors

```
sqrt(sum((pred1-testing$wage)^2))
```

```
[1] 827.1
```

```
sqrt(sum((pred2-testing$wage)^2))
```

```
[1] 866.8
```

```
sqrt(sum((combPred-testing$wage)^2))
```

```
[1] 813.9
```

Predict on validation data set

```
pred1V <- predict(mod1,validation); pred2V <- predict(mod2,validation)
predVDF <- data.frame(pred1=pred1V,pred2=pred2V)
combPredV <- predict(combModFit,predVDF)
```

Evaluate on validation

```
sqrt(sum((pred1V-validation$wage)^2))
```

```
[1] 1003
```

```
sqrt(sum((pred2V-validation$wage)^2))
```

```
[1] 1068
```

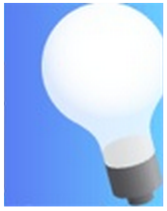
```
sqrt(sum((combPredV-validation$wage)^2))
```

```
[1] 999.9
```

Notes and further resources

- Even simple blending can be useful
- Typical model for binary/multiclass data
  - Build an odd number of models
  - Predict with each model
  - Predict the class by majority vote
- This can get dramatically more complicated
  - Simple blending in caret: caretEnsemble (use at your own risk!)
  - Wikipedia ensemlbe learning

Recall - scalability matters



Innovation
by Mike Masnick
Fri, Apr 13th 2012
12:07am

5

Why Netflix Never Implemented The Algorithm That Won The Netflix $1 Million Challenge

from the *times-change* dept

You probably recall all the excitement that went around when a group **finally won** the big Netflix $1 million prize in 2009, improving Netflix's recommendation algorithm by 10%. But what you might *not* know, is that **Netflix never implemented that solution itself**. Netflix recently put up a blog post **discussing some of the details of its recommendation system**, which (as an aside) explains why the winning entry never was used. First, they note that they *did* make use of an earlier bit of code that came out of the contest:

http://www.techdirt.com/blog/innovation/articles/20120409/03412518422/

http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html