Combining predictors

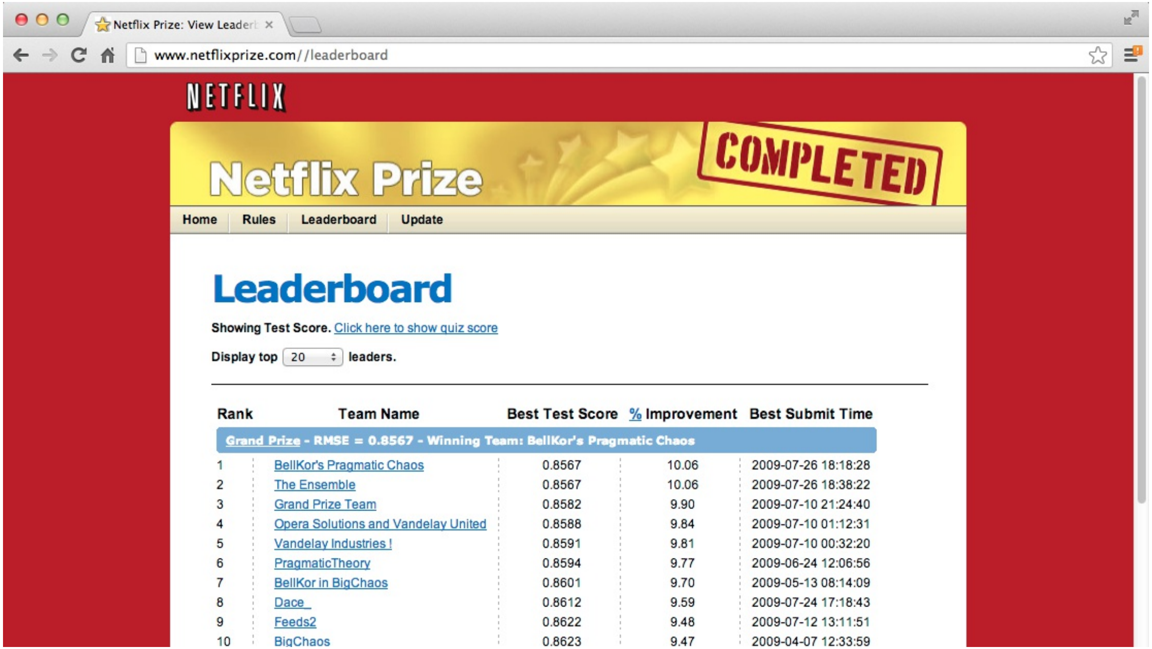
Jeffrey Leek, Assistant Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Key ideas

- You can combine classifiers by averaging/voting `can combine very different classifiers`
- 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 `107 classifiers!  (not just predictor variables)`



Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59

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:

wrong:
nicht im Quadrat!

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

With 101 independent classifiers

- 99.9% majority vote accuracy

??

warum sollten die Classifier unabhaengig sein?

Wie kommt diese Formel zustande?

--> see explanation from Forum on the right:

Forum: https://class.coursera.org/predmachlearn-005/forum/thread?thread_id=157

Assume your classifiers are independent weighted coin tosses, where 70% of the times it lands on heads, 30% tails.

Flip the weighted coin 5 times, and choose Heads if the majority of times it lands on heads, Tails if the majority is Tails. The accuracy of the majority-vote classifier is equivalent to the probability that the majority vote is heads. This can only happen in three conditions, leading to this formula:

```
$$P(Majority \ is\ Heads) = P(3\ Heads, 2\ Tails) + P(4\ Heads, 1\ Tail) + P(All\ Heads)$$
```

Each one of those probabilities can be calculated with the binomial pmf (dbinom in R)

```
dbinom(3, 5, .7) + dbinom(4,5,.7) + dbinom(5,5,.7)
```

Each of these terms correspond with the ones in the slide.

This returns

```
[1] 0.83692
```

Sounds amazing when I first saw it in the video lecture, but you're right about the independence assumption. Most classifiers are deterministic and therefore are conditionally dependent on data, where as a coin flip is always independent.

Approaches for combining classifiers

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

Text

Example with Wage data

"Model stacking"

Create training, test and validation sets

```
library(ISLR); data(Wage); library(ggplot2); library(caret);
Wage <- subset(Wage,select=-c(logwage))        remove logwage: that would be a too easy predictor

# 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)
```

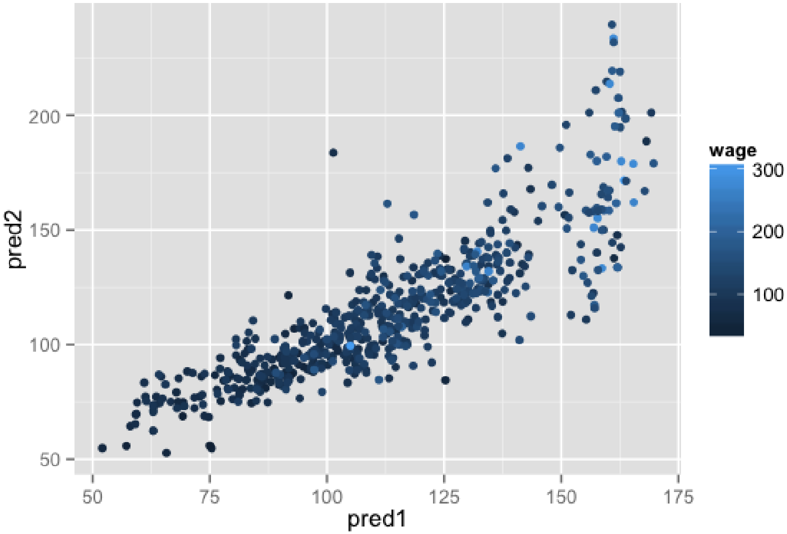
```
[1] 898  11
```

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)
```



-> the 2 models are close
   but don't exactly fit.

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)
```

combine predictions from model 1 and model 2 in one dataframe

Then train a model based on those 2 predictions

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
```

combined model has lower error

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              again, combined model has lower error, even on the validation dataset
```

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:

`–> computational complexity!`

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

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