Practical Machine Learning

Yato

March 24, 2017

Part 2 What's prediction

Components of a predictor and the relative order of importance

question > input data > features > algorithm > parameters > evaluation

An important point

The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data.—-John Tukey

Garbage in = Garbage out

1. May be easy (movie ratings -> new movie ratings)
2. May be harder (gene expression data -> disease)
3. Depends on what is a "good prediction".
4. Often more data > better models
5. The most important step!

Features matter!

Properties of good features

- Lead to data compression
- Retain relevant information
- Are created based on expert application knowledge

Common mistakes
* Trying to automate feature selection
* Not paying attention to data-specific quirks
* Throwing away information unnecessarily

Prediction is about accuracy tradeoffs

- Interpretability versus accuracy
- Speed versus accuracy
- Simplicity versus accuracy
- Scalability versus accuracy

04 In sample versus out of sample

In Sample Error: The error rate you get on the same data set you used to build your predictor. Sometimes called resubstitution error.
Out of Sample Error: The error rate you get on a new data set. Sometimes called generalization error.

- Key ideas:
  - Out of sample error is what you care about
  - In sample error < out of sample error
  - The reason is overfitting
- Matching your algorithm to the data you have

05 Prediction study design

Prediction study design

1. Define your error rate
2. Split data into: Training, Testing, Validation (optional)
3. On the training set pick features: Use cross-validation
4. On the training set pick prediction function: Use cross-validation
5. If no validation: Apply 1x to test set
6. If validation: (1) Apply to test set and refine and (2) Apply 1x to validation.

Avoid small sample sizes

- Suppose you are predicting a binary outcome
  - Diseased/healthy
  - Click on ad/not click on ad
- One classifier is flipping a coin
- Probability of perfect classification is approximately:
  - $\left(\frac{1}{2}\right)^{sample\ size}$
  - $n = 1$ flipping coin 50% chance of 100% accuracy
  - $n = 2$ flipping coin 25% chance of 100% accuracy
  - $n = 10$ flipping coin 0.10% chance of 100% accuracy

Rules of thumb for prediction study design

- If you have a large sample size
  - 60% training
  - 20% test
  - 20% validation
- If you have a medium sample size
  - 60% training
  - 40% testing
- If you have a small sample size
  - Do cross validation
  - Report caveat of small sample size

Some principles to remember

- Set the test/validation set aside and *don't look at it*
- In general *randomly* sample training and test
- Your data sets must reflect structure of the problem
  - If predictions evolve with time split train/test in time chunks (calledbacktesting in finance)
- All subsets should reflect as much diversity as possible
  - Random assignment does this
  - You can also try to balance by features - but this is tricky

06 types of Errors

Basic terms

In general, **Positive** = identified and **negative** = rejected. Therefore:

True positive = correctly identified = Sick people correctly diagnosed as sick

False positive = incorrectly identified = Healthy people incorrectly identified as sick

True negative = correctly rejected = Healthy people correctly identified as healthy

False negative = incorrectly rejected = Sick people incorrectly identified as healthy.

Common error measures

1. Mean squared error (or root mean squared error)–Continuous data, sensitive to outliers
2. Median absolute deviation–Continuous data, often more robust. For continuous data:
3. Sensitivity (recall)–If you want few missed positives
4. Specificity–If you want few negatives called positives
5. Accuracy–Weights false positives/negatives equally
6. Concordance
7. Predictive value of a positive (precision)–When you are screeing and prevelance is low

07 Receiver Operating Characteristic (ROC)

- In binary classification you are predicting one of two categories
  - Alive/dead
  - Click on ad/don't click
- But your predictions are often quantitative
  - Probability of being alive
  - Prediction on a scale from 1 to 10
- The *cutoff* you choose gives different results

08 Cross Validation

1. Accuracy on the training set (resubstitution accuracy) is optimistic
2. A better estimate comes from an independent set (test set accuracy)
3. But we can't use the test set when building the model or it becomes part of the training set
4. So we estimate the test set accuracy with the training set.

Cross-validation Approach:

1. Use the training set
2. Split it into training/test sets (Random subsampling/K-fold/Leave one out)
3. Build a model on the training set
4. Evaluate on the test set
5. Repeat and average the estimated errors

Used for:

1. Picking variables to include in a model
2. Picking the type of prediction function to use
3. Picking the parameters in the prediction function
4. Comparing different predictors

Considerations

- For time series data data must be used in "chunks"
- For k-fold cross validation
  - Larger k = less bias, more variance
  - Smaller k = more bias, less variance
- Random sampling must be done *without replacement*
- Random sampling with replacement is the *bootstrap*
  - Underestimates of the error
  - Can be corrected, but it is complicated (0.632 Bootstrap)
- If you cross-validate to pick predictors estimate you must estimate errors on independent data.

10 Caret Package

Caret functionality

- Some preprocessing (cleaning)
  - preProcess
- Data splitting
  - createDataPartition
  - createResample
  - createTimeSlices
- Training/testing functions
  - train
  - predict
- Model comparison
  - confusionMatrix

Machine learning algorithms in R

- Linear discriminant analysis
- Regression
- Naive Bayes
- Support vector machines
- Classification and regression trees
- Random forests
- Boosting
- etc.

Covariate creation

Covariates are sometimes called predictors and sometimes called features.

Two levels of covariate creation: **Level 1: From raw data to covariate**

Level 2: Transforming tidy covariates

```
library(kernlab);data(spam)
spam$capitalAveSq <- spam$capitalAve^2
```

Level 1, Raw data -> covariates

- Depends heavily on application
- The balancing act is summarization vs. information loss
- Examples:
- Text files: frequency of words, frequency of phrases (Google ngrams), frequency of capital letters.
- Images: Edges, corners, blobs, ridges (computer vision feature detection)
- Webpages: Number and type of images, position of elements, colors, videos (A/B Testing)
- People: Height, weight, hair color, sex, country of origin.
- The more knowledge of the system you have the better the job you will do.
- When in doubt, err on the side of more features
- Can be automated, but use caution!

Level 2, Tidy covariates -> new covariates

- More necessary for some methods (regression, svms) than for others (classification trees).
- Should be done *only on the training set*
- The best approach is through exploratory analysis (plotting/tables)
- New covariates should be added to data frames

Notes and further reading

- Level 1 feature creation (raw data to covariates)
- Science is key. Google "feature extraction for [data type]"
- Err on overcreation of features
- In some applications (images, voices) automated feature creation is possible/necessary
  - http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf
- Level 2 feature creation (covariates to new covariates)
- The function *preProcess* in *caret* will handle some preprocessing.

- Create new covariates if you think they will improve fit
- Use exploratory analysis on the training set for creating them
- Be careful about overfitting!
- preprocessing with caret
- If you want to fit spline models, use the *gam* method in the *caret* package which allows smoothing of multiple variables.
- More on feature creation/data tidying in the Obtaining Data course from the Data Science course track.

preProcessing PCA

Basic PCA idea

- We might not need every predictor
- A weighted combination of predictors might be better
- We should pick this combination to capture the "most information" possible
- Benefits
- Reduced number of predictors
- Reduced noise (due to averaging)

We could rotate the plot

$$X = 0.71 \times \text{num415} + 0.71 \times \text{num857}$$

$$Y = 0.71 \times \text{num415} - 0.71 \times \text{num857}$$

Related problems

You have multivariate variables $X_1, \ldots, X_n$ so $X_1 = (X_{11}, \ldots, X_{1m})$

- Find a new set of multivariate variables that are uncorrelated and explain as much variance as possible.
- If you put all the variables together in one matrix, find the best matrix created with fewer variables (lower rank) that explains the original data.

The first goal is statistical and the second goal is data compression.

Related solutions - PCA/SVD

SVD

If X is a matrix with each variable in a column and each observation in a row then the SVD is a "matrix decomposition"

$$X = UDV^T$$

where the columns of U are orthogonal (left singular vectors), the columns of V are orthogonal (right singluar vectors) and D is a diagonal matrix (singular values).

PCA

The principal components are equal to the right singular values if you first scale (subtract the mean, divide by the standard deviation) the variables.

Final thoughts on PCs

- Most useful for linear-type models
- Can make it harder to interpret predictors
- Watch out for outliers!
- Transform first (with logs/Box Cox)
- Plot predictors to identify problems
- For more info see
- Exploratory Data Analysis
- Elements of Statistical Learning

Predicting with Trees

Key ideas

- Iteratively split variables into groups
- Evaluate "homogeneity" within each group
- Split again if necessary

Pros:

- Easy to interpret
- Better performance in nonlinear settings

Cons:

- Without pruning/cross-validation can lead to overfitting
- Harder to estimate uncertainty
- Results may be variable

Example Tree

http://graphics8.nytimes.com/images/2008/04/16/us/0416-nat-subOBAMA.jpg

---

Basic algorithm

1. Start with all variables in one group
2. Find the variable/split that best separates the outcomes
3. Divide the data into two groups ("leaves") on that split ("node")
4. Within each split, find the best variable/split that separates the outcomes
5. Continue until the groups are too small or sufficiently "pure"

Measures of impurity

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \ in \ Leaf \ m} \mathbb{1}(y_i = k)$$

Misclassification Error:

$$1 - \hat{p}_{mk(m)}; k(m) = \text{most; common; k}$$

* 0 = perfect purity * 0.5 = no purity

Gini index:

$$\sum_{k \neq k'} \hat{p}_{mk} \times \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} p_{mk}^2$$

- 0 = perfect purity
- 0.5 = no purity

http://en.wikipedia.org/wiki/Decision_tree_learning

Measures of impurity

Deviance/information gain:

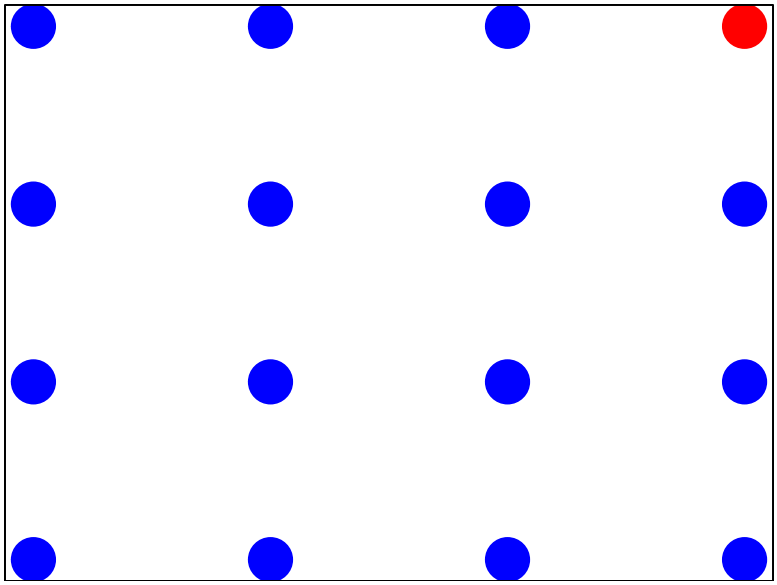
$$-\sum_{k=1}^{K} \hat{p}_{mk} \log_2 \hat{p}_{mk}$$

* 0 = perfect purity * 1 = no purity

http://en.wikipedia.org/wiki/Decision_tree_learning

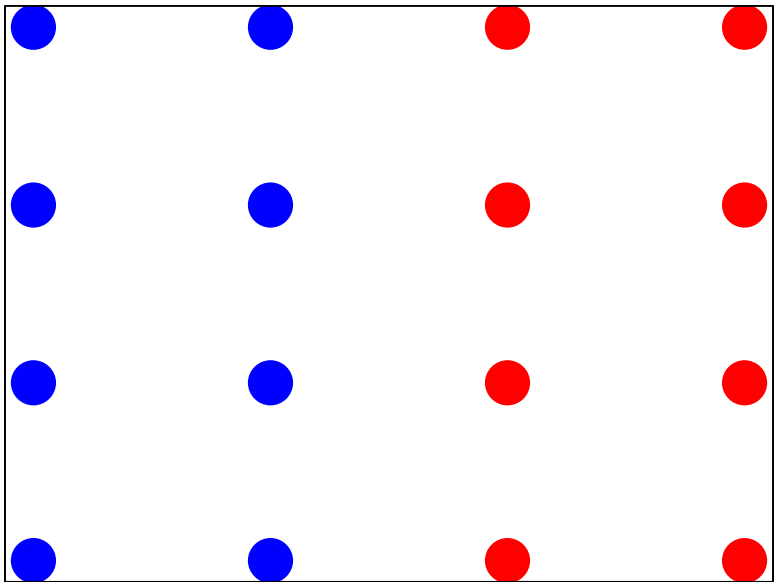
— &twocol w1:50% w2:50% ## Measures of impurity

*** =left



- **Misclassification:** $1/16 = 0.06$
- **Gini:** $1 - [(1/16)^2 + (15/16)^2] = 0.12$
- **Information:**$-[1/16 \times log2(1/16) + 15/16 \times log2(15/16)] = 0.34$

*** =right



- **Misclassification:** $8/16 = 0.5$
- **Gini:** $1 - [(8/16)^2 + (8/16)^2] = 0.5$
- **Information:**$-[1/16 \times log2(1/16) + 15/16 \times log2(15/16)] = 1$

Bootstrap aggregating (bagging)

Basic idea:

1. Resample cases and recalculate predictions
2. Average or majority vote

Notes:

- Similar bias
- Reduced variance
- More useful for non-linear function

Bagging in caret

- Some models perform bagging for you, in train function consider method options: bagEarth; treebag; bagFDA
- Alternatively you can bag any model you choose using the bag function

Random forests

1. Bootstrap samples
2. At each split, bootstrap variables
3. Grow multiple trees and vote

Pros:

1. Accuracy

Cons:

1. Speed
2. Interpretability
3. Overfitting

Random forests

http://www.robots.ox.ac.uk/~az/lectures/ml/lect5.pdf

Notes and further resources

Notes:

- Random forests are usually one of the two top performing algorithms along with boosting in prediction contests.
- Random forests are difficult to interpret but often very accurate.
- Care should be taken to avoid overfitting (see rfcv funtion)

Boosting

Basic idea

1. Take lots of (possibly) weak predictors
2. Weight them and add them up
3. Get a stronger predictor

Basic idea behind boosting

1. Start with a set of classifiers $h_1, \ldots, h_k$

- Examples: All possible trees, all possible regression models, all possible cutoffs.

2. Create a classifier that combines classification functions: $f(x) = \text{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.

- Goal is to minimize error (on training set)
- Iterative, select one h at each step
- Calculate weights based on errors
- Upweight missed classifications and select next h

Adaboost on Wikipedia

http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf

---

Simple example

http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf

---

Round 1: adaboost

http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf

---

Round 2 & 3

http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf

---

Completed classifier

http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf

---

Boosting in R

- Boosting can be used with any subset of classifiers
- One large subclass is gradient boosting
- R has multiple boosting libraries. Differences include the choice of basic classification functions and combination rules.
- gbm - boosting with trees.
- mboost - model based boosting
- ada - statistical boosting based on additive logistic regression
- gamBoost for boosting generalized additive models
- Most of these are available in the caret package
