

DATA MINING

Techniques and Challenges

Sam Steingold

<<http://sds.podval.org/data/dm.ps>>

Typical Problem

A large bank (e.g., Fidelity) wants to increase the profitability of its customer base by conducting cross-sell campaigns.

Contact selected customers ...

... with custom offers ...

... that they are likely to accept ...

... while increasing their profitability.

Methodology

- Split the input data set 4 ways:
 - train (balanced)
 - test (representative)
 - "train" and "test" are for model building
 - validate (representative)
 - for model validation and score -> probability conversion
 - hold (representative)
 - for cross-model and cross-target comparison
- Variable selection:
 - eliminate "irrelevant" variables
 - add combinations of "relevant" variables
- Train model(s?)
- Validate: performance on "validate" set should not be much worse than on train/test
- Compare valid models

Naive Bayesian Learning

Bayes, 1763:

$$P(A_i|C) = \frac{P(A_i)P(C|A_i)}{\sum_{i=1}^n P(A_i)P(C|A_i)}$$

C - discrete class, vector A - attributes with discrete values

Predict C=c from A=a when $P(C=c | A=a)$ is maximal

Assume that the components of A are independent (given C).

From Bayes:

$$P(C = c|A = a) = P(C = c) \frac{\prod_{i=1}^n P(A_i = a_i|C = c)}{P(A = a)}$$

"Maximum likelihood" probability estimate

$$P(A_i = a_i|C = c) = \frac{\text{count}(A = a \wedge C = c)}{\text{count}(C = c)}$$

Naive Bayesian Learning (math 1)

Two classes (0&1), a - attribute value vector for a test example,

$$b_0 = P(C = 0), b_1 = P(C = 1) = 1 - b_0$$

$$p_{j0} = P(A_j = a_j | C = 0), p_{j1} = P(A_j = a_j | C = 1)$$

Then (z - normalizing constant):

$$p = P(C = 1 | A = a) = b_1 \prod_{j=1}^k p_{j1} / z$$

$$q = P(C = 0 | A = a) = b_0 \prod_{j=1}^k p_{j0} / z = 1 - p$$

$$\log p - \log q = \sum_{j=1}^k (\log p_{j1} - \log p_{j0}) + \log b_1 - \log b_0$$

Let $w_j = \log p_{j1} - \log p_{j0}$, and $b = \log b_1 - \log b_0$

Naive Bayesian Learning (math 2)

Then the log odds

$$\log \frac{1-p}{p} = - \sum_{j=1}^k w_j - b$$

$$p = \frac{1}{1 + e^{-\sum_{j=1}^k w_j - b}}$$

In general, let each $A(j)$ take $v(j)$ values, and let

$$w_{jj'} = \log P(A_j = a_{jj'} | C = 1) - \log P(A_j = a_{jj'} | C = 0)$$

$$P(C(x) = 1) = \frac{1}{1 + e^{-\left(\sum_{j=1}^k \sum_{j'=1}^{v(j)} I(A_j(x)=a_{jj'}) w_{jj'}\right) - b}}$$

Here I is the indicator function, i.e., $I(X)=1$ if X is true and $I(X)=0$ if X is false.

Bayesian Classifier

- Equivalent to a perceptron with a sigmoid activation function and sparse encoding of attributes (a separate input for each possible value of each attribute).
- A non-parametric, non-linear extension of logistic regression, which, in turn, is equivalent to a perceptron with a sigmoid activation function and a single input node for each attribute (attribute values are encoded with their magnitude).

Boosting (Freund & Schapire, 1995)

- Learn several classifiers
- Each classifier up-weights previously misclassified examples
- The final classifier outputs a weighted sum of all the classifiers

Elkan, 1997:

- Boosted Naive Bayesian classifier is representationally equivalent to a multilayer perceptron with a single hidden layer.

Advantages

- Low computational complexity: $O(T \cdot e \cdot f)$ where
 - T is the number of boosting rounds
 - e is the number of examples
 - f is the number of attributes (features)

- Real-time model updates

- Interpretable results
 - offers new statistically significant actionable insights

- Excellent performance
 - Fast training
 - Fast scoring
 - Good model quality

Disadvantages

- The assumption of attribute independence is usually false
- The score returned by the model is usually either almost 0 or almost 1
 - Use selective classifier (keep a few best attributes)
 - Use special magic to convert scores to probability estimates
- Requires discrete attributes, thus necessitates a binning process
 - Best is "equal height" binning
 - Slow ($e \cdot \log(e)$), compared to the actual training
 - Complicates real-time adaptive modeling: may require periodic re-binning
- Neural networks and SVM often exhibit (insignificantly) better model quality (at the price of slow training)

Model quality: Hit rate (accuracy)

Defined as the fraction of correct predictions:

$$A = \frac{\text{count}(\textit{Prediction} = \textit{Class})}{\text{count}(\textit{examples})}$$

- Often meaningless, especially when the targets are scarce.
 - "San Diego weather forecast"

- Loses information - collapses the score (real from 0 to 1) into the boolean prediction (maybe appropriate only if we will contact the whole population)

- Ignores the different value of errors and hits
 - false positives (wasted phone call or harassed traveler)
 - false negatives (lost sale or customer, school bus blown up)
 - true positives (profit made, terrorist arrested)

How are predictive models used?

The usual procedure:

- Sort the examples by their model score
- Contact the top 5% (or 10% or 20%) customers

The cut-off is determined by the values of errors and hits ("cost/benefit matrix") and the percentage of targets in the sample ("base rate") with the goal of profit maximization.

But what if the values are not known in advance?

Model quality: Lift curve

Let Cumulative Percent Hits $CPH(M,p)$ be the percentage of targets in the first $p\%$ of the list sorted by decreasing score of model M . Define

$$\text{lift}(M, p) = \frac{CPH(M, p)}{p}$$

- "lift curve" is the CPH vs percentile graph
 - (AKA receiver operating characteristic - ROC curve)

- If model A lift curve dominates (is strictly above) model B lift curve, then for any fixed cost/benefit matrix and for any p , the benefit of selecting the top p observations using the model A will be higher than the benefit of selecting the top p observations using B.

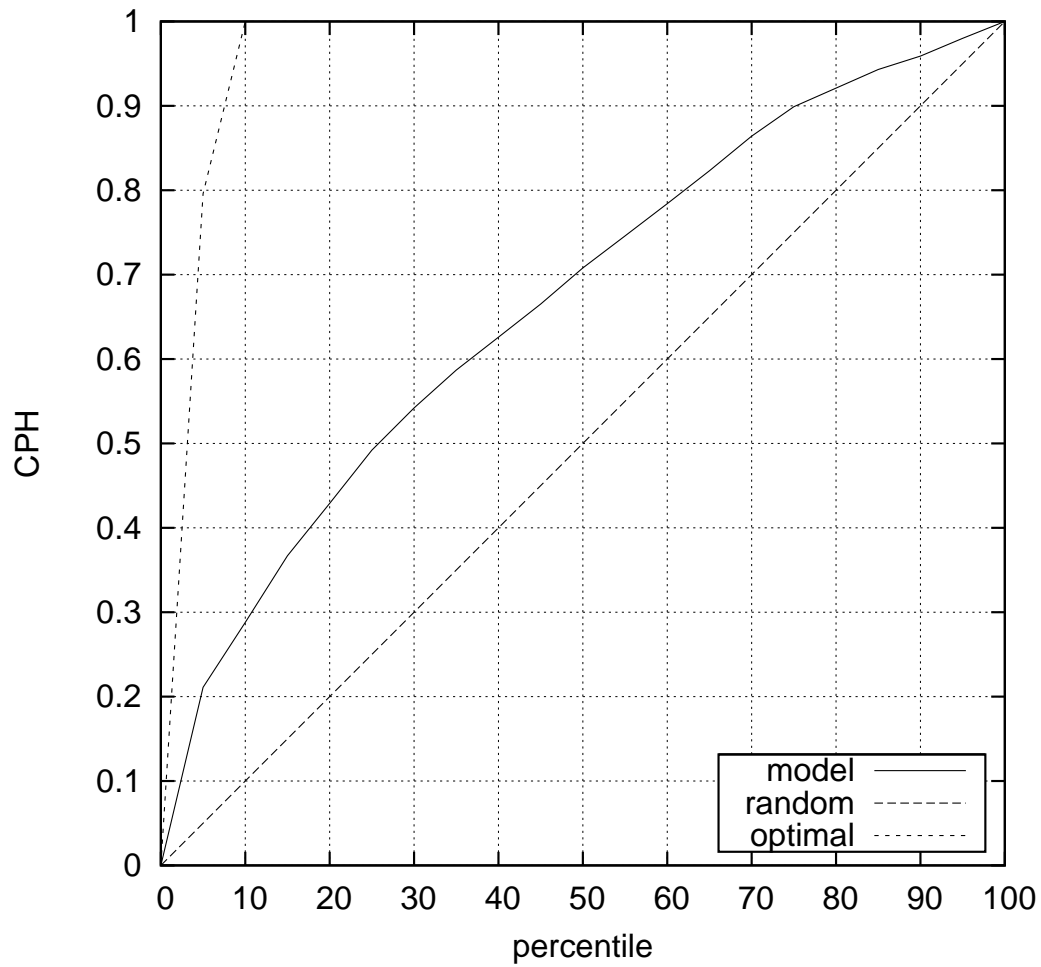
Lift Table example

%	recs	hits	hits%	lift	model CPH	optimal CPH
5	1045	277	26.51	4.22	0.211	0.796
10	2090	378	18.09	2.88	0.288	1
15	3135	481	15.34	2.44	0.367	1
20	4180	563	13.47	2.15	0.429	1
25	5225	646	12.36	1.97	0.492	1
30	6270	711	11.34	1.81	0.542	1
35	7315	770	10.53	1.68	0.587	1
40	8360	821	9.82	1.56	0.626	1
45	9405	872	9.27	1.48	0.665	1
50	10450	929	8.89	1.42	0.708	1
55	11495	979	8.52	1.36	0.746	1
60	12540	1029	8.21	1.31	0.784	1
65	13585	1080	7.95	1.27	0.823	1
70	14630	1134	7.75	1.23	0.864	1
75	15675	1180	7.53	1.20	0.899	1
80	16720	1208	7.22	1.15	0.921	1
85	17765	1237	6.96	1.11	0.943	1
90	18810	1258	6.69	1.07	0.959	1
95	19855	1286	6.48	1.03	0.980	1
100	20900	1312	6.28	1	1.000	1

legend:

- % - the percentile (= CPH of the "random" model)
- recs - the number of records
- hits - the number of targets correctly identified
- hits% - the percentage of hits, which is the number of hits divided by the number of records
- lift - the lift, equal to hits%/b
- model CPH - the cumulative percent hits for the given model
- optimal CPH - the cumulative percent hits for the ideal model

Lift Curve example



Sum of Cumulative Percent Hits

$$\text{SumCPH}(M) = \int_0^1 \text{CPH}(M, p) dp$$

$$\text{SumCPH}(\text{random}) = 1/2$$

$$\text{SumCPH}(\text{ideal}) = 1-b$$

SumCPH of a predictive model will be between these two values

If $\text{SumCPH} < 1/2$, just use the opposite of the model!

SumCPH is bad!

- SumCPH is not intuitive - the meaning of 0.69 is not clear
- SumCPH by itself does not tell us how close it is to the maximum
- The higher the base rate, the lower is the maximum possible SumCPH and the measure of lift quality should account for it

Lift quality

Normalize SumCPH so that the ideal model will be 1 and the random 0:

$$\begin{aligned} \text{L-quality}(M) &= \frac{\text{SumCPH}(M) - \text{SumCPH}(\textit{Random})}{\text{SumCPH}(\textit{Best}) - \text{SumCPH}(\textit{Random})} \\ &= \frac{2\text{SumCPH}(M) - 1}{1 - b} \end{aligned}$$

L-quality(random) = 0

L-quality(ideal) = 1

L-quality of a predictive model will be between these two values:

L-quality(above) = 35.6%

Cut-off choice

max(profit), assuming constant profit matrix

$$\begin{aligned}\text{profit}(p) &= \text{CPH}(p)bN \times \text{TP} \\ &+ (p - \text{CPH}(p)b)N \times \text{FP} \\ &+ (1 - \text{CPH}(p))bN \times \text{FN} \\ &+ [(1 - p) - b(1 - \text{CPH}(p))]N \times \text{TN}\end{aligned}$$

$$\text{profit}'(p) = 0$$

$$\begin{aligned}\text{profit}'(p)/N &= \text{CPH}'(p)b \times \text{TP} \\ &+ (1 - \text{CPH}'(p)b) \times \text{FP} \\ &- \text{CPH}'(p)b \times \text{FN} \\ &- (1 - \text{CPH}'(p)b) \times \text{TN}\end{aligned}$$

$$\text{CPH}'(p) = \frac{\text{TN} - \text{FP}}{b(\text{TP} - \text{FP} - \text{FN} + \text{TN})}$$

Problems - Methodological

Overfitting

- occurs when the model is chosen because of high accuracy on the training set, which actually arises from a statistically unreliable pattern in it
- that is what the "validation" set is for
 - if too few targets for a "validation" set, use regularization (e.g., early stopping or averaging)

Information Leakers

- dependent variables with causal relationships with the target
- cannot be used for predictive modeling
- need a domain expert to be identified
- eliminating leakers reduces model quality

Problems - Technical

Too few targets (e.g., 300 out of 100,000)

- use weights to emulate a balanced data set
- "wiggle" the targets

Too many columns ("the dimensionality curse")

- Combine/Drop attributes
- Roll-up ("time-based" --> "individual-based")
- Use SVM

Data integrity (inconsistencies, invalid entries etc)

- Study the data before modeling
- Check for (in?)sane values

Gotchas

Suppose we have 1 target for 1,000 samples (base rate $b=0.1\%$).

Model lift at $p=1\%$ is 100 (stunning performance!)

Then in the top 1% there will be 10% targets.

I.e., 90% will be false alarms!

Randomness IS there!

No machine learning algorithm can achieve perfect performance!

More Gotchas

Suppose you test positive for a very rare illness (1 case per 10,000 people). The test is 99.99% accurate - if you do have the illness, it is certain to detect it, and the probability of a false positive is just 0.01%. What are the odds that you are ill?

Even!

I = Ill, T = Test positive

$$\begin{aligned} P(I|T) &= \frac{P(I)P(T|I)}{P(I)P(T|I) + P(\neg I)P(T|\neg I)} \\ &= \frac{0.0001 \times 1}{0.0001 \times 1 + 0.9999 \times 0.0001} \\ &\approx \frac{1}{2} \end{aligned}$$

In other words, two individuals out of 10,001 test positive - the ill one and the one who is the false positive, and you are

Cross-model comparison (binomial)

Null hypothesis: $\text{CPH}(20\%) = 15\% = p$.

Suppose we have 4,000 examples, then the expected number of targets in the top 20% set is $t = 120 = 800 * 15\%$ out of $n = 800 = 4,000 * 20\%$.

The anticipated standard deviation = $\sqrt{np(1-p)} = 10.1$.

To reject the null hypothesis with a confidence of over 95%, a model needs to score over 2 standard deviations away from the expected number, i.e., as long as the number of targets is from 100 to 140, the model is statistically indistinguishable from the null hypothesis model.

Cross-model comparison (chi-squared)

Compare the performance of two models on a data set neither used in training.

Let A be the number of examples classified correctly by the first model but misclassified by the second one and B vice versa.

$$s = \frac{(|A - B| - 1)^2}{A + B}$$

Under the null hypothesis that both models are equally accurate, the McNemar statistic s is approximately chi-squared with 1 degree of freedom (mean 1, standard deviation is $\sqrt{5}$)

Conclusions

The randomness is there, so using a learning algorithm without an intuitive understanding of the underlying statistic is futile.

The dangers are many and tough (overfitting, leakers...)

Therefore the methodology is crucial.

Thank You!

References

- C. Elkan "Boosting and Naive Bayesian Learning", UCSD, Technical Report CS97-557, 1997
- Y. Freund and R.E. Schapire "A decision-theoretic generalization of of on-line learning and an application to boosting", Proceedings of the Second European Conference on Computational Learning Theory, 1995, 23-37
- G. Piatetsky-Shapiro, S. Steingold "Measuring Lift Quality in Database Marketing", SIGKDD Explorations, Vol. 2:2, (2000), 81-86
- T.G. Dietterich "Approximate statistical tests to comparing supervised classification learning algorithms", Neural Computation, 10(7), 1895-1924, 1998

The End
