Feature extraction – a probabilistic approach

Ondřej Sýkora

November 18, 2010

Introduction

- Today, we will...
  - ...follow up with what Zuzka started last week, this time from a probabilistic point of view
  - ...have a short overview of the NIPS challenge
  - ...review several methods for feature extraction, that are based on the probability theory

Motivation

- The goal of feature extraction
  - For a classification problem on a given dataset, find a reasonably small (smallest) set of features that does not give significantly worse classification results than the complete set of features.
- Why feature extraction?
  - Fewer features may lead to: faster classification, faster learning, better generalization, easier obtaining of data, the data use less space on disk, ...
- The definition is somewhat ambiguous
  - Let's see, how it was implemented in the NIPS Challenge

NIPS Challenge 2003

- An international challenge in feature extraction
- The task: (binary) classification on 5 datasets with different characteristics
  - Very high dimensionality (500-100 000 features)
    - ...especially when compared to the numbers of samples
  - Pre-processing: *probes*
    - irrelevant (random) variables
    - Independent on the class ⇒ should be removed by good feature extraction algorithms
  - The data sets were splitted into three sets:
    - Training: class labels available to the participants
    - Validation: class labels not available, immediate response to the submitted data via challenge website
    - Test: class labels not availabe, used to evaluate the participants at the end of the competition

NIPS Challenge 2003

Datasets

Dataset	Features	Trn + Val + Tst
Arcene	10 000	100 + 100 + 700
Dexter	20 000	300 + 300 + 2000
Dorothea	100 000	800 + 350 + 800
Gisette	5 000	6000 + 1000 + 6500
Madelon	500	2000 + 600 + 1800

NIPS Challenge 2003
Evaluation

- Evaluation metrics:
  - $BER = \frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$
  - F_{feat} = fraction of features selected by the classifier (self-reported)
  - F_{prob} = fraction of probes selected
- Tournament: each classification competes with each other
  - If BER of the two classifiers are significantly different (McNemar test, $\alpha = 0.05$), the better one wins
  - If the difference of F_{feat} is greater than 0.05, the lower one wins
  - If the difference of F_{prob} is greater than 0.05, the lower one wins
  - The algorithms are equally good
  - Winner gets 1 point, loser gets -1; in case of draw, both get 0

NIPS Challenge 2003

Results

Method	Group	Chapter	Score	BER (Rk)	AUC (Rk)	Ffeat	Fprob
BayesNN-DFT	Neal/Zhang	10	71.43	6.48 (1)	97.20 (1)	80.3	47.77
BayesNN-large	Neal	10	66.29	7.27 (3)	96.98 (3)	60.3	28.51
BayesNN-small	Neal	10	61.14	7.13 (2)	97.08 (2)	4.74	2.91
final_2-3	Chen	12	49.14	7.91 (8)	91.45 (25)	24.91	9.91
BayesNN-large	Neal	10	49.14	7.83 (5)	96.78 (4)	60.3	28.51
final2-2	Chen	12	40	8.80 (17)	89.84 (29)	24.62	6.68
Ghostminer1	Ghostminer	23	37.14	7.89 (7)	92.11 (21)	80.6	36.05
RF+RLSC	Torkkola/Tuv	11	35.43	8.04 (9)	91.96 (22)	22.38	17.52
Ghostminer2	Ghostminer	23	35.43	7.86 (6)	92.14 (20)	80.6	36.05
RF+RLSC	Torkkola/Tuv	11	34.29	8.23 (12)	91.77 (23)	22.38	17.52

Bayesian Classification

- For a sample x, select the most probable hypothesis $H^* = \arg\max_H P(H|x)$
  - Hypothesis H_i: sample x belongs to class C_i
  - Problem: in most cases, $P(H|x)$ is unknown, difficult to obtain
- Bayes rule:

$$P(H|x) = \frac{P(x|H)P(H)}{P(x)} = \frac{P(x|H)P(H)}{\sum_H P(x|H)P(H)}$$

- $P(x|H)$, $P(H)$ and $P(x)$ might be known, or easier to measure

Bayesian Classification

Naive Bayes Classifier

- Special case: $x = (x_1, x_2, \dots, x_n)$, and x_i are conditionally independent given H:
  $P(x|H) = P(x_1, x_2, \dots x_n|H) = P(x_1|H)P(x_2|H)\dots P(x_n|H)$
- Classification using:

$$P(H|x) = \frac{P(x_1|H)P(x_2|H)\dots P(x_n|H)P(H)}{\sum_H P(x_1|H)P(x_2|H)\dots P(x_n|H)P(H)}$$

- Advantage: very simple and efficient representation
- Disadvantage: conditional independence is a very strong precondition
  - Often used with good results even if the conditional independence does not hold
- For binary classification: only calculate $P(positive|x)$, compare it to a threshold (e.g. 50 %)

Selective Naive Bayes Classifier

- Deals with the problem of redundant or strongly correlated variables
- Question: how do we find them?
  - The do not improve classification results!
- Algorithm:
  1. Start with an empty set of features F
  2. Train Naive Bayes classifier, only using features from F, measure its accuracy on the training set
  3. For each feature f not in F:
     - Train Naive Bayes classifier using features $F \cup \{f\}$, measure accuracy on the training set
  4. Select the feature f^* that most improves accuracy of the model; halt if adding any feature not in F degrades the accuracy
  5. Add f^* to F, go to step 2
- Performance comparable to Naive Bayes or better

Enhanced Selective Naive Bayes Classifier

with Optimal Discretization

- Improved version of Selective NB for the NIPS Challenge
- A different evaluation criterion:
  - Accuracy replaced by AUC (Area Under lift Curve)
    - Plot $\frac{TP}{TP+FP+TN+FN}$ against $\frac{TP+FP}{TP+FP+TN+FN}$ for all values of threshold (remember NBC for binary classification)
    - More sensitive than accuracy
- When the feature selection halts, the optimal threshold is found using the lift curve
  - the threshold with maximal value of $\frac{TP}{TP+FP+TN+FN}$

Dec. 8^{th}	ESNB+NN challenge entry					The winning challenge entry					
Dataset	Score	BER	AUC	Feat	Probe	Score	BER	AUC	Feat	Probe	Test
OVERALL	-28	12.42	93.12	1.04	1.43	71.43	6.48	97.20	80.3	47.77	1

Enhanced Selective Naive Bayes Classifier

With Optimal Discretization!

- MODL - Bayes Optimal discretization
  - Discretization model:
    - There are m samples with J different class labels
    - The class labels of the samples are ordered according to the value of the variable being discretized
    - The labels are separated into I intervales; Interval I_i contains m_i labels, $m_{i,j}$ of them are labels for class j.
  - Optimal discretization – maximizes $P(discretization|data)$
  - Three-stage prior – distribution over discretization models
    - The number of intervals I is uniformly distributed between 1 and m
    - For a given number of intervals I, each partitioning of the samples to the intervals is equiprobable
    - The distributions of class labels in each interval are independent of each other

Enhanced Selective Naive Bayes Classifier

With Optimal Discretization!

- Given models distributed according to the three-stage prior, the Bayes-optimal one minimizes

$$\log(m) + \log\binom{m+I-1}{I-1} + \sum_{i=1}^{I} \log\binom{m_i+J-1}{J-1} +$$

$$\sum_{i=1}^{I} \left(\frac{m_i!}{m_{i,1}! \ldots m_{i,J}!} \right)$$

- To find the model, use a greedy bottom-up strategy

Input Variable Importance Definition based on Empirical Data Probability Distribution

- A wrapper method, calculates importance based on difference in classification results
- A classifier is treated as a black box or a mapping f from samples to target values
- Notation:
  - $f(x) = f(x_1, x_2, \ldots, x_n)$ – output of the prediction model on sample x
  - V_{ij} – value of j-th attribute of i-th sample
  - $P_{V_j}(v)$ – marginal probability distribution over the values of j-th feature in the data
  - $P_x(u)$ – distribution of the samples
  - $f_j(x; b) = f(x_1, \ldots, x_{j-1}, b, x_{j+1}, \ldots, x_n)$

Input Variable Importance Definition based on Empirical Data Probability Distribution

- Define importance of variable j for prediction model f

$$
\begin{aligned}
S(V_j|f) &= \int\int |f(u) - f_j(u;v)| dP_x(u) dP_{V_j}(v) \\
&= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{m}\sum_{i=1}^{m}|f(x_i) - f_j(x_i;V_{kj})|
\end{aligned}
$$

- Start with all features, in each step remove variable j with minimal importance; re-train the model after removing each variable
- In the NIPS Challenge, used with multilayer perceptron NN

Dec. 1^{st}	Our best challenge entry						The winning challenge entry				
Dataset	Score	BER	BER*	AUC	Feat	Probe	Score	BER	AUC	Feat	Probe
Overall	-62.18	16.37	20.1	83.63	1.12	21.47	88.00	6.84	97.22	80.3	47.8

Bayesian Neural Networks

Bayesian Learning

- Suppose we have:
  - A classification model $P(Y|X, \theta)$ with parameter vector θ
  - Set of training samples X_{train} with class labels Y_{train}
- A classical approach is to choose such θ that maximizes $P(Y_{train}|X_{train}, \theta)$
- Now suppose we have a prior distribution $P(\theta)$. Then, we can use the Bayes rule:

$$P(\theta|Y_{train}, X_{train}) = \frac{P(\theta)P(Y_{train}|X_{train}, \theta)}{\int P(\theta)P(Y_{train}|X_{train}, \theta)d\theta}$$

- Remember, we have a model $P(Y|X,\theta)$, prior $P(\theta)$ and posterior distribution $P(\theta|Y_{train}, X_{train})$
- Together, we get a new model:

$$P(Y_{new}|X_{new}, X_{train}, Y_{train}) =$$

$$= \int P(Y_{new}|X_{new}, \theta) P(\theta|X_{train}, Y_{train}) d\theta$$

- For fixed X_{train} and Y_{train}, we have a classifier $P(Y_{new}|X_{new})$
- So far, so good, but where is feature selection?

Bayesian Neural Networks

An Example with Logistic Regression

- Consider the logistic model

$$P(Y = c|X = x) = \frac{1}{1 + e^{-(\alpha+\beta^T x)}}, \theta = (\alpha, \beta^T) \in R^{n+1}$$

- How do we get the prior distribution $P(\theta)$?
  - A classical solution: we make one up!
  - The prior is not that important, a reasonable one should work well
  - α and the elements of β should be independent
  - For α, any broad enough distribution should be fine
  - For β, use $N(0, \sigma^2 I_n)$, where σ is another parameter
  - With such choice of β, the probability distribution is spherical around 0 $\rightarrow$ the output of the model is invariant to orthonormal transformations of x and β

Bayesian Neural Networks

An Example with Logistic Regression

- Consider the logistic model

$$P(Y = c|X = x) = \frac{1}{1 + e^{-(\alpha + \beta^T x)}}, \theta = (\alpha, \beta^T) \in R^{n+1}$$

- How do we get the prior distribution $P(\theta)$?
  - Now consider using $N(0, diag(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2))$
  - Let $\sigma_i^2 = 0$ for some i. Then the value of β_i is forced to 0, effectively eliminating the i-th feature
  - Analogically, using small σ_j^2 for a feature j forces small values of β_j, giving a hint about the relevance if the j-th feature
  - However, this introduces new parameters σ_i^2 that needs a value
  - Use a reasonable prior and determine them using the Bayes rule and maximum likelihood

Bayesian Neural Networks
Practical implementation

- The same approach can be used for bayesian learning of other models
- For neural networks, use weights as the parameter vector
  - Hyperparameters σ_i^2 can be used for weights to the input neurons
- Practical notes
  - Lots of complex mathematics, how to deal with that
    - compute analytically whathever can be computed that way
    - use numerical solvers for other integrals
    - use Monte-Carlo Markov Chain for sampling of parameters
    - create multiple instances of the classification model with sampled parameters to produce the final result – "Bagging done right"

Conclusions

Results

Method	Group	Chapter	Score	BER (Rk)	AUC (Rk)	Ffeat	Fprob
BayesNN-DFT	Neal/Zhang	10	71.43	6.48 (1)	97.20 (1)	80.3	47.77
BayesNN-large	Neal	10	66.29	7.27 (3)	96.98 (3)	60.3	28.51
BayesNN-small	Neal	10	61.14	7.13 (2)	97.08 (2)	4.74	2.91
final_2-3	Chen	12	49.14	7.91 (8)	91.45 (25)	24.91	9.91
BayesNN-large	Neal	10	49.14	7.83 (5)	96.78 (4)	60.3	28.51
final2-2	Chen	12	40	8.80 (17)	89.84 (29)	24.62	6.68
Ghostminer1	Ghostminer	23	37.14	7.89 (7)	92.11 (21)	80.6	36.05
RF+RLSC	Torkkola/Tuv	11	35.43	8.04 (9)	91.96 (22)	22.38	17.52
Ghostminer2	Ghostminer	23	35.43	7.86 (6)	92.14 (20)	80.6	36.05
RF+RLSC	Torkkola/Tuv	11	34.29	8.23 (12)	91.77 (23)	22.38	17.52
FS+SVM	Lal	20	31.43	8.99 (19)	91.01 (27)	20.91	17.28
Ghostminer3	Ghostminer	23	26.29	8.24 (13)	91.76 (24)	80.6	36.05
CBAMethod3E	CBA group	22	21.14	8.14 (10)	96.62 (5)	12.78	0.06
CBAMethod3E	CBA group	22	21.14	8.14 (11)	96.62 (6)	12.78	0.06
Nameless	Navot/Bachr.	17	12	7.78 (4)	96.43 (9)	32.28	16.22

- Questions, comments?