FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Aprendizagem Automática

Trabalho Prático 1
2016

Parametrizar, ajustar e comparar os classificadores **Logistic Regression**, ***K-Nearest Neighbours*** e ***Naive Bayes***

Professores:
Ludwig Krippahl
Susana Nascimento

Alunos:
Bruno Ferreira - 44763
Emídio Correia - 46815
David Gago - 41710

Índice

Introdução

O trabalho consistia em distinguir notas falsas de notas verdadeiras baseando-nos nos fatores: variância(variance), obliquidade (skewness), curtosetransformada Wavelet da imagem da nota (curtosis of Wavelet Transformed image) e entropia (entropy). Sendo esses dados obtidos através de um .csv e estando separados por vírgulas em que os primeiros quatro campos correspondem aos fatores e o final é uma etiqueta para representar se a nota é verdadeira, com valor 0, ou se é falsa, valor 1.

A programação foi feita em Python 2.7 pois aparenta ser uma boa linguagem para dados científicos, como PhD Cyrille Rossant nos conta em
http://cyrille.rossant.net/why-using-python-for-scientific-computing/
Python é grátis e open source.
Em python a manipulação dos dados é mais fácil que noutras linguagens.
Python inclui muitas bibliotecas especializadas, as quais usámos para a regressão logística, k-NN e separação de dados em diferentes conjuntos.

Implementação do Naive Bayes classifier

Criámos uma classe (kdeNB) para implementar o naive Bayes. Definimos os métodos fit, score e predict, que são os métodos necessários nos outros classificadores. Segue-se uma breve explicação de cada um.

fit(X,Y) - função que calcula e guarda $C^{Naive\ Bayes} = ln\, p(C_k) + \sum_{j=1}^{N} ln\, p(x_j|C_k)$.

score(X,Y) - compara a lista de valores previstos com a lista de valores esperados e devolve a métrica accuracy_score.

predict(X) - para cada elemento de X, calcula a probabilidade de este ser 0 ou 1, devolve a lista dos valores previstos.

Os parâmetros Bandwidth, K e C

Os três parâmetros otimizados foram Bandwidth, K e C.

O parâmetro Bandwidth, vai influenciar a "largura" da *Kernel function* , e para diferentes valores de *h* , obtemos diferentes classificadores:

$$\hat{y}(x) = \frac{1}{h}\sum_{t=2}^{N}(x^t - x^{t-1})K\left(\frac{x - x^t}{h}\right)y^t$$

O valor de *h* varia de 0.01 a 1, com incrementos de 0.02.

Em relação ao K, é um parâmetro usado no classificador *K-NN,* e determina o(s) ponto(s) do *training set* mais próximos do novo ponto a serem usados no classificador. Um valor baixo para este parâmetro significa que usaremos poucos pontos o que implicará grandes variações na previsão. Um valor alto implica poucas variações na previsão, mas com a inclusão de dados que poderão estar longe de serem relevantes.
Os valores de K variam de 1 até 39, só com valores ímpares.

O parâmetro C, é usado na classe *LogisticRegression*. É usado para prevenir o *overfitting*.
Definimos o nosso parâmetro C tal como no enunciado, começando com C = 1 e a duplicar a cada iteração (20 iterações).

Estimativa de erro de cada classificador

Pelo facto de estarmos interessados em minimizar o erro esperado em relação a qualquer elemento do universo de dados, isto é, o *true error* ,e não só o erro obtido através do nosso conjunto de dados de treino (*trainning error*), é necessário reservar alguns elementos dos nossos dados especificamente para estimar este erro. Estes elementos não serão usados na obtenção do melhor C, k ou bandwidth, utilizando apenas o conjunto de treino separado em conjunto de treino e validação. Assim sendo, divide-se os dados em três conjuntos: o de treino para fazer o *fit* de cada modelo, o de validação para estimar o comportamento da nossa função aquando da previsão de dados, fora do conjunto de treino, para obter o erro estimado e obter a melhor hipótese, e o de teste para estimar o *true error.*

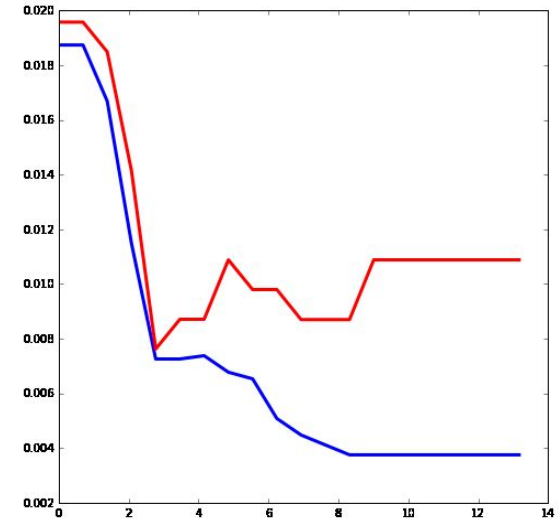

Fig 1 - *Logistic Regression, erro de treino e validação (linha azul e vermelha respectivamente)*

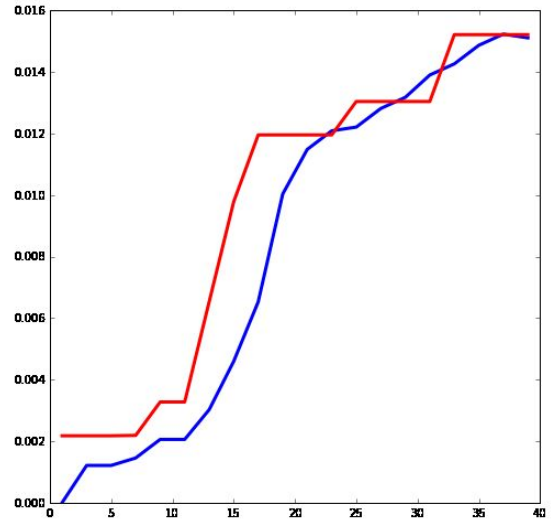

Fig 2 - *K-NN, erro de treino e validação (linha azul e vermelha respectivamente)*

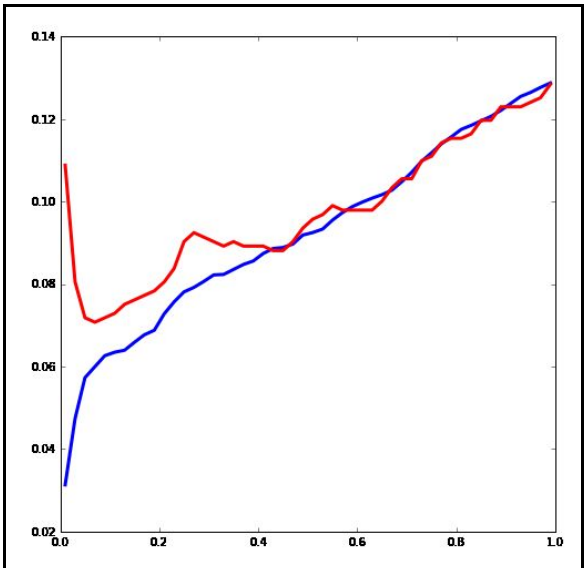

Fig 3 - Naive Bayes, erro de treino e validação *(linha azul e vermelha respectivamente)*

Taxa de erro do conjunto de teste (*True error*):

LogReg = 0.0132450331126
C = 16

kNN = 0.00220750551876
k = 5

NB = 0.0662251655629
bw = 0.07

McNemar's Test

Estimativa do verdadeiro valor de cada classificador comparado com outro classificador, usando o teste McNemar. Este método tem como base o cálculo:

$$\frac{(|e_{01} - e_{10}| - 1)}{e_{01} + e_{10}} \approx \chi_1^2$$

Sendo e_{01} o número de valores em que o primeiro classificador classificou mal e o segundo classificou bem e e_{10} o número de valores em que o segundo classificador classificou mal e o primeiro classificou bem.

Se o valor calculado for superior a 3.84 (valor padrão para 95% confiança) podemos rejeitar as hipóteses nulas (com 95% de confiança), isto é, quando os dois classificadores têm uma performance idêntica.

No nosso caso estamos interessados nos seguintes classificadores:

- LogReg: Regressão logística (Logistic Regression)
- kNN: K-nearest neighbours
- NB: Naive Bayes

Obtivemos os seguintes resultados para cada um dos classificadores:

1. **LogReg vs kNN** = 2.29
2. **kNN vs NB** = 27.03
3. **NB vs LogReg** = 14.69

No 1º caso concluímos que não há grandes diferenças entre estes classificadores, pois o resultado do teste McNemar deu abaixo de 3.84.

Houve grandes diferenças nos classificadores usados no 2º e 3º caso, pois o resultado do teste McNemar está acima do valor de comparação 3.84. Isto significa que os classificadores não se vão comportar da mesma maneira nesta aplicação.

Conclusão

O classificador que apresentou menor taxa de erro para este trabalho foi o k-nearest neighbours. Daí concluímos que para identificar as notas falsas, o classificador mais fiável seria o k-nearest neighbours

Para esta aplicação, tanto regressão logística como o k-NN apresentam resultados semelhantes.

O classificador naive bayes não se adequa à aplicação pois tem um erro demasiado alto comparado com os outros dois (regressão logística e k-NN) e o resultado do teste McNemar com estes foi demasiado alto.