Universidade de São Paulo

SCC0231

Instrodução a Sistemas Inteligentes

Projeto 1

Alunos:	*Número USP:*
Matheus Araujo Jorge	9266705
Gabriel Martins da Silva	9266747

8 de Novembro de 2018

Conteúdo

1 Introdução

Este projeto teve por objetivo principal, comparar vários classificadores utilizando como base de dados o conjunto de textos *cbr-ilp-ir*, onde cada texto é composto pelo título e pelo *abstract* de um artigo, e as classes dos textos são justamente *cbr*, *ilp* e *ir*.

Para executar os algoritmos foram necessárias técnicas de pré-processamento de texto como remoção de *stopwords*, radicalização e cortes de Luhn. Foi feita uma comparação da influência do valor dos cortes no resultado final de cada algoritmo utilizado.

Aproveitou-se ainda para testar se as suposições feitas em dois classificadores de Bayes distintos influenciam o resultado dos classificadores, no contexto do conjunto de dados citado.

O código completo da análise pode ser encontrado em [1].

2 Teoria

2.1 Classificador de Bayes

Um classificador de Bayes [3] utiliza o conhecido Teorema de Bayes para estimar as probabilidades de um determinado objeto com atributos $\mathbf{X} = \mathbf{x}$ pertencer a uma dada classe $\mathbf{Y} = \mathbf{k}$, utilizando a probabilidade da ocorrência daquele conjunto de atributos na classe k. Matematicamente, o Teorema de Bayes é dado como segue:

$$P_r(Y = k|\mathbf{X} = \mathbf{x}) = \frac{P_r(\mathbf{X} = \mathbf{x}\ |Y = k)\ P_r(Y = k)}{P_r(\mathbf{X} = \mathbf{x})} \tag{1}$$

O classificador cálcula a probabilidade para todas as classes, e classifica o objeto como pertecendo à classe k se a prabalidade para essa classe for a maior de todas. Em termos matemáticos:

$$\hat{k} = \underset{k}{\operatorname{argmax}}\ P_r(Y = k|\mathbf{X} = \mathbf{x}) \tag{2}$$

Como o denominador da expressão 1 é comum a todas as classes, ele pode ser desconsiderado quando estamos maximizando as probabilidades, resultanto então em:

$$\hat{k} = \underset{k}{\operatorname{argmax}}\ P_r(\mathbf{X} = \mathbf{x}\ |Y = k)\ P_r(Y = k) \tag{3}$$

A parte mais complicada é estimar o valor de $P_r(\mathbf{X} = \mathbf{x}\ |Y = k)$, uma vez que seu cálculo muitas vezes é computacionalmente inviável. Para isso são feitas algumas suposições a respeito de seu comportamento.

2.2 Naive Bayes

No caso de Naive Bayes [3], a suposição feita é que todos os atributos são independentes. Apesar de ser raramente verdade, essa suposição apresenta normalmente resultados satisfatórios. Temos então que a classe é dada por:

$$\hat{k} = \underset{k}{\operatorname{argmax}}\ P_r(Y = k) \prod_{i=1}^{n} P_r(x_i|Y = k) \tag{4}$$

onde n é o número de atributos.

2.3 Análise Discriminante Linear

No caso da análise discriminante linear(LDA) [2] a suposição é que, para cada classe, os dados seguem uma distribuição normal multivariada e que todas essas distribuições compartilham a mesma matrix de covariâncias Σ e tem vetores de médias μ_k diferentes. Temos então:

$$\hat{k} = \underset{k}{\operatorname{argmax}} \; P_r(Y = k) \; f_k(\mathbf{x}) \tag{5}$$

onde,

$$f_k(\mathbf{x}) = \frac{1}{2^{n/2} \; |\Sigma|^{1/2}} \exp(-\frac{(x - \mu_k)^T \; \Sigma^{-1} \; (x - \mu_k)}{2}) \tag{6}$$

3 Experimentos

3.1 Ferramentas

Para a realização deste projeto, foi utilizada a linguagem de programação Python e suas bibliotecas. A biblioteca *os* foi usada para acesso aos textos. Para manipulação dos dados foram usadas as bibliotecas *pandas* e *numpy*. Para cálculos, *scipy* e *math* foram as escolhidas. Para construção do modelo vetorial, a biblioteca *nltk* foi usada para extração de *tokens* e utilização de *stopwords*, enquanto a *scikit-learn* foi usada para cálculo de **td-idf**. Por fim, a *scikit-learn* também foi utilizada para a criação dos modelos com os quais o algoritmo implementado foi comparado.

Os experimentos foram em ambiente Linux, utilizando a distibuição **Manjaro** de 64 *bits*, com ambiente gráfico *XFCE*. A máquina possui um processador *Intel i7-8550u*, com frequência de 1.8 Ghz e memória RAM de 16 GB.

3.2 Pré-processamento

O pré-processamento dos dados seguiu os procedimentos padrões de processamento de texto. Inicialmente, foram extraídos os *tokens* de cada texto. Nos *tokens* extraídos foi aplicado um filtro, onde foram selecionados aqueles que continham no mínimo uma letra(excluindo números literais, por exemplo), com comprimento de no mínimo dois caracteres e que não estivessem na lista de *stopwords*, obtida em *nltk.corpus.stopwords.words('english')*. Essas *stopwords* são genéricas para a língua inglesa. Procurou-se uma lista com *stopwords* específicas para a classificação de artigos científicos, mas não foi encontrada nenhuma e, portanto, continuamos com a lista genérica. Após a aplicação do filtro foi utilizada um radicalizador, também da biblioteca *nltk*. O algoritmo escolhido foi o Porter Stemming [4].

Optou-se por utilizar a representação conhecida como **tf-idf** em todos os algoritmos. Para criar esse modelo, a partir dos *tokens* radicalizados, utilizamos a classe **TfidfVectorizer**, do módulo *sklearn.feature_extraction.text*.

3.3 Implementação do Naive Bayes

Como foi decidido utilizar a representação citada acima, foi necessária uma implementação do Naive Bayes diferente da que foi exposta na proposição do projeto. O modo encontrado [3], foi supor que cada variável segue uma distribuição normal univariada, dentro de cada uma das classes existentes.

3.4 Demais algoritmos

Para algoritmos de comparação foram escolhidos os seguintes:

- Análise Discriminante Linear (*LDA*);
- SVM(*Support Vector Machines*), com dois *kernels* diferentes:
  - Linear e
  - Sigmoid.

3.5 Metodologia

Inicialmente, variou-se os valores utilizados no corte de Luhn, utilizando os seguintes valores:

- Inferior: 0.0, 0.1 e 0.2;
- Superior: 1.0, 0.9 e 0.8.

Para todos os algoritmos, foi executada uma rodada do **10-fold cross-validation** para cada valor de corte, de onde foram obtidos 10 valores de acurácia para cada algoritmo. Os resultados apresentados na seção 5, são as médias dos 10 valores obtidos.

Quisemos também testar, se as suposições feitas nos classificadore de Bayes(Naive Bayes e LDA) influenciavam nos resultados. Para isso, escolhemos as configuração com valores de corte de Luhn de inf = 0.2 e sup = 1.0(melhor resultado para ambos), e executamos 10 rodadas do *10-fold cross-validation*. Fizemos isso pois, para testar estatisticamente se as duas médias são iguais, devemos ter um número razoável de pontos. O teste realizado foi o *teste t de Student*.

4 Resultados

Nesta seção, apresentaremos os resultados descritos no experimentos descritos. Abaixo está apresentada a Tabela 4, com as acurácias médias de cada algoritmo para cada uma das configurações do corte de Luhn.

Inferior	Superior	Algoritmo			
		NB	LDA	SVM-L	SVM-S
0.0	1.0	0.942	0.710	0.991	0.480
0.0	0.9	0.942	0.708	0.991	0.480
0.0	0.8	0.940	0.707	0.991	0.480
0.1	1.0	0.934	0.908	0.986	0.480
0.1	0.9	0.935	0.903	0.986	0.480
0.1	0.8	0.928	0.894	0.986	0.480
0.2	1.0	0.962	0.974	0.984	0.480
0.2	0.9	0.962	0.970	0.984	0.480
0.2	0.8	0.956	0.968	0.982	0.480

Tabela 1: Acurácia dos diferentes algoritmos, para diferentes valores do corte de Luhn.

Para o segundo experimento, o Naive Bayes apresentou acurácia média de 96.1 % enquanto o LDA apresentou 97.2 % de acurácia. A hipótese que queremos testar é que acurácias são iguais, e que, portanto, as suposições feitas não afetam a qualidade do algoritmo de um modo geral para esses dados. Primeiro, verificamos se as variâncias de cada conjunto eram iguais, pois isso pode afetar no teste utilizado. Obtivemos que a variância do Naive Bayes foi 0.006 e a do LDA foi 0.005. Foi considerado então que elas eram iguais, e foi aplicado o teste mencionado, considerando duas variáveis independentes. O teste obteve um valor -3.114, com *p-value* igual a 0.001.

5 Discussão

Pela Tabela 4, podemos perceber que o algoritmo que apresenta melhores resultados sempre é o SVM com *kernel* linear. Em contrapartida, o SVM com *kernel* sigmoid é o algoritmo com pior resultado em todos os testes. Isso indica uma clara dependência do *kernel* na qualidade da classificação. Além disso, também podemos supor que os dados apresentados são melhores separados por uma função linear do que pela sigmoid. Isso pode não ser verdade para outro conjunto de dados ou se fosse feito outro tipo de pré-processamento. Por fim, ainda tratando dos algoritmos SVM, outra suposição que pode ser feita é que a quantidade de amostras é insuficiente para treinar o algoritmo com *kernel* sigmoid, enquanto para o *kernel* linear essa quantidade de dados é absolutamente suficiente.

Analisando agora os dois classificadores de Bayes, observamos um fenômeno interessante: quando não eliminamos as palavras com frequência muito baixa em documentos, o Naive Bayes se sai muito melhor do que o LDA. Entretanto, aumentando o valor apenas um pouco, temos um salto considerável na acurácia do LDA. Esse fato indica que as variáveis de menor frequência fazem com que a suposição de normalidade multivariada assumida pelo LDA seja ferida. Quando as retiramos, a suposição se torna mais acertada e, com isso, os resultados do algoritmo melhoram consideravelmente.

Quando não retiramos os atributos citados no parágrafo anterior, vemos claramente que o Naive Bayes é melhor do que o LDA. Entretanto, quando o fazemos os dois classificadores bayseanos se comportam de maneira muito parecida. Para dizer se um é realmente melhor que o outro, foi feito um teste estatítico, com os resutados apresentados na seção anterior. Pelo valor do *p-value* encontrado, podemos dizer, com 99 % de certeza, que as médias dos dois algoritmos não são iguais. Agora resta saber qual é a média maior. O valor de estatística negativa indica que o segundo conjunto de dados apresentado à função tem média maior do que o primeiro. Neste caso, o segundo conjunto corresponde aos dados do LDA e, portanto, podemos afirmar que, estatisticamente, o LDA apresenta resultados melhores do que o Naive Bayes para a configuração escolhida.

6 Conclusão

Através dos resultados obtidos, pode-se constatar que a escolha de parâmetros dos algoritmos é um dos passos fundamentais de uma boa análise. Parâmetros como o *kernel* do SVM e a escolha dos valores do corte de Luhn, mostraram-se de grande significância para o resultado final obtido. Além disso, é importante conhecer a teoria por traz de seus classificadores para identificar possíveis causas de erros, como foi o caso das variáveis de baixa frequência no LDA. Quando variáveis que atrapalhavam as suposições do modelo foram retiradas, foi estatisticamente comprovado que o LDA se saiu melhor do que o Naive Bayes

Tratando dos resultados propriamente ditos, o SVM com *kernel* linear foi o melhor algoritmo. Porém, o Naive Bayes e o LDA obtiveram resultados bem semelhantes e com uma complexidade muito menor. Muitas vezes é mais interessante escolher os algoritmos mais simples, pois estes apresentam menos parâmetros que são necessários estimar e desse modo possuem uma convergência(se satisfeitas as condições do algoritmo) mais rápida.

Referências

[1] Código fonte. `https://github.com/matheusjorge/TextMining_Proj1`. Acessado em 06 de Novembro de 2018.

[2] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[3] T. M. Mitchell. Generative and discriminative classifiers: Naive bayes and logistic regression, chapter 1, 2005.

[4] M. Porter. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14, 03 1980.