Implementação de Kernel Customizado Aplicado à Análise de Sentimentos em Resenhas de Filmes

Luciana dos Santos Kayo e Paulo Mei

Orientador: Professor Doutor Marco Dimas Gubitoso

Instituto de Matemática e Estatística – Universidade de São Paulo

IME-USP

Introdução

O aprendizado supervisionado é uma área da aprendizagem de máquina (*Machine Learning*) que busca encontrar uma função a partir da análise de um conjunto dados de treino, que possa ser usada em novos conjuntos com informações semelhantes à aquelas separadas para o treinamento.

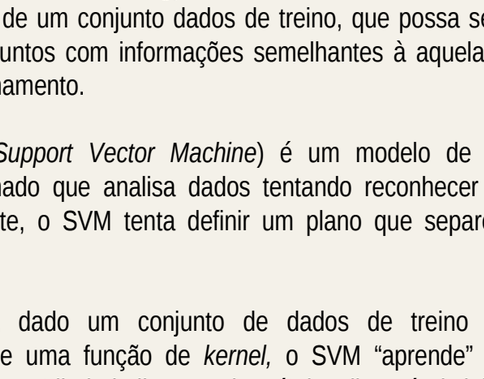
O SVM (*Support Vector Machine*) é um modelo de aprendizado supervisionado que analisa dados tentando reconhecer um padrão. Basicamente, o SVM tenta definir um plano que separe pontos no espaço.

Para isso, dado um conjunto de dados de treino previamente rotulados, e uma função de *kernel,* o SVM "aprende" a classificar novos dados, atribuindo-lhes um dos rótulos disponíveis inicialmente.

Uma das aplicações do SVM é no processamento de linguagem natural para análise de sentimentos. A análise de sentimentos, também conhecida como mineração de opinião, é um ramo da mineração de dados que visa encontrar informação subjetiva em textos, ou seja, que tem por objetivo determinar o sentimento do autor quando ele escreveu determinada afirmação.

Este trabalho foi inspirado na competição "*When Bag of Words Meets Bags of Popcorn*", lançada por meio do Kaggle, em dezembro de 2014. A idéia original da competição era utilizar a ferramenta Word2Vec para análise de sentimentos em resenhas de filmes.

Com base nessa premissa, optou-se pelo estudo de diferentes algoritmos e kernels a fim de compará-los quanto à sua acurácia na determinação da polaridade de uma resenha, ou seja, na classificação da resenha em positiva ou negativa.



Tecnologias Utilizadas

A linguagem de programação escolhida para a implementação do trabalho foi o Python. Python é uma linguagem de alto nível, amplamente empregada em algoritmos de SVM e procesamento de linguagem natural. Além disso, um grande número de ferramentas e bibliotecas disponíveis para trabalhar com aprendizagem de máquina e processamento textual são escritas em Python, ou integram-se facilmente à linguagem. Algumas delas, que foram utilizadas no código, estão brevemente descritas a seguir.

O Pandas é uma biblioteca para manipulação e análise de dados que oferece uma estrutura, chamada data frame, com indexação integrada, e métodos a ela aplicáveis, tais como *merge* e *join*.

O NLTK (*Natural Language Toolkit*) é uma ferramenta dedicada a trabalhar com linguagem natural que possui diversas bibliotecas que permitem o emprego de técnicas como classificação, tokenização, tagging, entre outras.

O Scikit-Learn é uma ferramenta para mineração de dados, baseada em NumPy, SciPy e matplotlib, que oferece algoritmos para calssificação, como o SVM, e opções de pré-processamento dos dados que serão submetidos a esses algoritmos.

Implementação

Os dados utilizados para o estudo são provenientes da base do IMDb (*Internet Movie Database*), e estavam disponíveis para download na página do Kaggle sobre a competição mencionada anteriormente.

Cada dado é uma linha, com um identificador, um rótulo e a resenha. O rótulo foi aplicado com o seguinte critério: se a nota do filme, baseada em todos as resenhas dadas, era maior ou igual a 7, então a resenha foi rotulada como positiva e seu rótulo tem o valor 1; porém se a nota estava abaixo de 5, então a resenha recebeu a classificação de negativa e o rótulo 0.

Parte da base rotulada foi utilizada como conjunto de treino, que serviria para definir a função de classificação. Os dados restantes, por também conterem rótulos, foram usados para fazer a validação cruzada, obtendo-se assim a porcentagem de acerto da função na classificação das resenhas.

O pré-processamento das resenhas deu-se pela limpeza dos dados obtidos do Kaggle, removendo-se primeiramente as tags de HTML e as palavras vazias (ou *stop-words*), como "the", "at", "is", considerando-se que as resenhas estão em inglês. Em seguida foram separados apenas os adjetivos e advérbios, preservando-se sua ordem de ocorrência no texto. As classes de palavras desejadas foram obtidas após a aplicação da versão mais simples do processo de POS Tagging (*Part-of-Speech Tagging*), que separa palavras em adjetivos, substantivos, verbos e advérbios.

Os dados limpos foram submetidos a dois tipos diferentes de *kernels* para o algortimo de SVM. O primeiro *kernel* usado é linear e estava disponível como parte do Scikit-Learn. De uma maneira simplificada, eles procura definir o classificador com base numa função linear. Para esses *kernels* o vetor de características foi determinado utilizando-se as técnicas de *Bag of Words* e Tf-idf, cujo foco é a frequência de termos no texto.

O segundo *kernel*, idealizado pelo professor Gubi, foi customizado e pré-computado antes do uso no SVM. Foi implementado a patir de uma adaptação do algoritmo presente em [1], também chamado de "*Gap-weighted subsequence*", considerando-se sequências de palavras, ao invés de apenas *substrings*.

O algoritmo original procura similaridade entre palavras considerando que podem existir "espaços" entre as *substrings*. O exemplo abaixo tenta simplificar a idéia por trás do *kernel*.

Considerando-se duas palavras, por exemplo, "barco" e "banco", e um tamanho pré-fixado de 2 letras para as *substrings* a serem estudadas, o algoritmo consideraria, por exemplo, que há semelhança entre as palavras devido a presença tanto da *substring* "b-o", quanto da "ba". No entanto a *substring* "ba" terá um peso maior, por serem letras imediantamente consecutivas na palavra, e a *substring* "b-o"terá um peso menor, pois o *kernel* também leva em conta a distância entre as letras.

A versão customizada faz uma análise bem parecida, porém ao invés de olhar cada palavra, letra a letra, a resenha inteira é analisada, palavra a palavra, considerando, é claro, as condições de que apenas adjetivos e advérbios foram utilizados. O exemplo a seguir explica, também de forma resumida, como o algortimo foi adaptado.

Considera-se três resenhas: "O filme é muito bom", "Muito engraçado e bom" e "Muito bom esse filme". Após o processamento inicial do texto, restariam, para cada resenha, "muito bom", "muito engraçado bom" e "muito bom". Apesar de todas as resenhas conterem as palavras "muito"e "bom", a primeira e a última seriam consideradas mais similares, pois existe um espaço de uma palavra na segunda resenha.

Uma vez escolhidos os *kernels* que seriam utilizados, as amostras de dados foram testadas e submetidas aos dois processos. O resultado foi organizado com base no tempo de execução e na porcentagem de acerto na classificação após a aplicação da validação cruzada.

Resultados

As duas primeiras tabelas mostram o resultado do SVM linear com uso de *Count Bag of Words* e *Tf-idf Bag of Words* para uma amostra de 25 mil resenhas. Devido ao altíssimo custo computacional do kernel customizado, foi necessário reduzir a amostra para mil resenhas, na terceira tabela.

SVM Count Bag of Words			
	Sem stop-words	Só adjetivos	Adjetivos + Advérbios
Score-in	0.9963	0.8403	0.8996
Score-out	0.8284	0.7955	0.8205

SVM Tf-idf Bag of Words			
	Sem stop-words	Só adjetivos	Adjetivos + Advérbios
Score-in	0.9394	0.8359	0.8815
Score-out	0.8732	0.7981	0.8377

SVM com Kernel Customizado	
	Adjetivos + Advérbios
Score-in	0.9967
Score-out	0.5275

Tempos de execução			
	Sem stop-words	Só adjetivos	Adjetivos + Advérbios
Limpeza*	17m 20.79s	10h 00m 35.22s	10h 01m 03.41s
SVM Count Bag of Words	22m 26.75s	07m 49.56s	14m 33.00s
SVM Tf-idf Bag of Words	23m 06.03s	05m 35.15s	09m 38.79s
SVM Customizado**	-	-	1h 42m 10.42s

(*) Limpeza para 25 mil resenhas
(**) Tempo de execução para mil resenhas

Conclusão

Filtrar por adjetivos mantém muito da informação buscada e reduz em muito a dimensão do vetor de características, porém a um custo muito alto.
Utilizando-se adjetivos e advérbios, obtém-se uma pequena melhora no resultado e não é necessário gastar um tempo muito maior com essa abortagem.
Com relação ao *kernel* customizado, havia a hipótese de que ele ficasse menos custoso com a redução da amostra analisada. No entanto, é possível concluir que, mesmo com a redução da amostra, o tempo necessário para computar a similaridade entre os reviews ainda é grande, e obtém-se um resultado ainda abaixo do esperado.

Referências

[1] http://www.jmlr.org/papers/volume2/lodhi02a/lodhi02a.pdf
https://www.kaggle.com/c/word2vec-nlp-tutorial
http://www.nltk.org/
http://scikit-learn.org/stable/
http://pandas.pydata.org/
http://wiki.python.org.br/
https://en.wikipedia.org/wiki/Support_vector_machine

Contato

Página Web:

http://www.linux.ime.usp.br/~lkayo/mac0499/
http://www.linux.ime.usp.br/~paulomei/mac0499/
http://www.ime.usp.br/~gubi

E-mail:
lkayo@linux.ime.usp.br
paulomei@linux.ime.usp.br
gubi@ime.usp.br