

Classificando Tweets – MAC5832

Pedro Henrique Rocha Briel

phrb@ime.usp.br

nUSP: 7143336

DCC - IME

Universidade de São Paulo

São Paulo, 15 de Maio de 2016

1 Introdução

Este relatório apresenta visualizações e algoritmos de aprendizado de máquina utilizados no primeiro *dataset* da disciplina **MAC5832**.

O *dataset* consiste de um conjunto de *tweets* sobre companhias aéreas, associado a avaliações de *sentimento*. Os algoritmos de aprendizado de máquina implementados foram o *Perceptron* e o *Gradient Descent*. Os dois algoritmos estão descritos nas Seções 3.1 e 3.2. Visualizações do *dataset* são apresentadas na Seção 2. A Seção 4 discute os resultados e conclui este relatório.

1.1 Estrutura de Diretórios

Os diretórios deste trabalho estão organizados da seguinte forma:

`./doc` Contém os arquivos necessários para gerar este documento.

`./data` Contém os conjuntos de dados disponibilizados para treinamento e testes.

`./src` Contém o código implementado para o *Perceptron* e *Gradient Descent*.

`./src/plot` Contém o código para gerar as figuras com as visualizações, e também as figuras.

2 Visualizando o *dataset*

Esta Seção apresenta alguns gráficos que ajudaram a visualizar o *dataset* utilizado no exercício.

A Figura 1 apresenta quatro gráficos gerados a partir dos exemplos do *dataset*. A Figura 1a associa o *sentimento total* a cada uma das companhias aéreas. O sentimento total é a soma dos valores de `airline_sentiment` dados

a cada *tweet*. É possível ver que considerar a companhia aérea como parte do modelo não trará muita informação, pois o sentimento total é negativo para todas as companhias, e não existem grandes variações de uma companhia para outra, com exceção da *Virgin*.

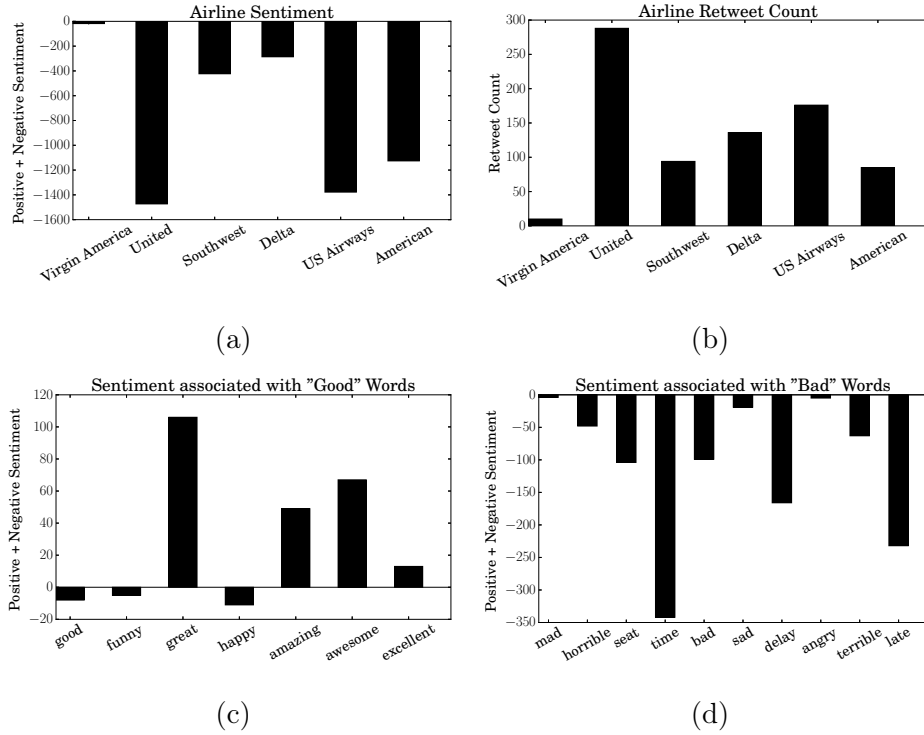


Figura 1: Visualizando o *dataset*

A Figura 1b associa a quantidade de *retweets* a cada companhia aérea. Novamente a *Virgin* se destaca das demais, dessa vez por ter poucos *retweets*. Ainda assim, não existe variação significativa na quantidade de *retweets* de cada companhia, e decidi por não usar essa característica no modelo.

As Figuras 1c e 1d mostram, respectivamente, os sentimentos totais associados a um conjunto com palavras consideradas “boas” ou positivas, e a outro com palavras consideradas “ruins” ou negativas. Com essas visualizações é possível concluir que usar as palavras contidas nos *tweets* como características nos vetores de características é uma boa ideia para o modelo, pois o sentimento total associado ao grupo de palavras positivas também

foi positivo, e o sentimento total associado ao grupo de palavras negativas também foi negativo.

3 Algoritmos de Aprendizado de Máquina

Esta Seção discute os algoritmos de aprendizado de máquina e suas implementações, feitas em `Python 3` e usando os módulos `nlTK`, `sklearn`, `numpy`, `csv` e `re`.

Utilizei o tutorial sobre *Bag of Words*¹, disponibilizado na página da Tarefa I, para gerar um *vocabulário* com as palavras mais frequentes nos *tweets* do *dataset*.

O vocabulário foi então usado para gerar os *vetores de características* utilizados para treinar os algoritmos e realizar as predições. Os vetores de características contêm um 1 na posição correspondente a uma determinada palavra se o *tweet* contém aquela palavra, e um 0 caso contrário.

Além disso, cada vetor de características é inicializado com um 1 na primeira posição, correspondente ao *threshold* do classificador.

3.1 Perceptron

Considere um vetor \mathbf{w} de pesos inicializado com $N + 1$ 1s, onde cada peso $w_i, 0 \leq i \leq N$ corresponde a uma característica dos vetores de características dos *tweets* do *dataset*, e N é o número de características extraídas do conjunto.

A cada passo do algoritmo *Perceptron*, vamos calcular a classificação c_j de cada um dos M vetores $\mathbf{x}_j, 0 \leq j \leq M$ de características usando a seguinte equação:

$$c_j = \text{step}\left(\sum_{i=0}^N w_i \mathbf{x}_j^i\right) = \text{step}\left(\mathbf{w}^T \mathbf{x}_j\right)$$

¹<https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words>

Onde a função *step* é dada por:

$$step(y) = \begin{cases} 1 & \text{se } y \geq 0 \\ -1 & \text{caso contrário.} \end{cases}$$

Depois, para cada vetor $\mathbf{x}_j, 0 \leq j \leq M$ vamos atualizar o vetor \mathbf{w} de pesos da seguinte maneira:

$$\mathbf{w} = \mathbf{w} + \alpha \mathbf{x}_j (z_j - c_j)$$

Aqui, \mathbf{x}_j é um vetor de características, α é a *taxa de aprendizado*, z_j é a classificação correta de \mathbf{x}_j e c_j é a predição feita pelo algoritmo nessa iteração.

A minha implementação do *Perceptron* executa esse procedimento para todos os exemplos no conjunto de testes até que o número de exemplos classificados erroneamente seja menor ou igual a um limite μ . Nesse *dataset*, utilizei $\mu = 6$. Depois do treinamento do modelo, obtemos um vetor \mathbf{w}^* , que é usado para gerar as predições para cada *tweet* contido nos dados para teste.

3.2 Gradient Descent

Considere um vetor \mathbf{w} de pesos inicializado com $N + 1$ 1s, onde cada peso $w_i, 0 \leq i \leq N$ corresponde a uma característica dos vetores de características dos *tweets* do *dataset*, e N é o número de características extraídas do conjunto.

A cada passo do algoritmo *Gradient Descent*, vamos calcular a classificação c_j de todos os vetores $\mathbf{x}_j, 0 \leq j \leq M, \mathbf{x}_j \in \mathbf{X}$ de características, onde \mathbf{X} é a matriz de exemplos. Faremos a classificação usando a seguinte equação:

$$\mathbf{C} = \mathbf{w}^T \mathbf{X}$$

Aqui, \mathbf{C} é um vetor contendo um *valor real* para cada exemplo $\mathbf{x}_j \in \mathbf{X}$. Para obter uma predição da classificação de algum \mathbf{x}_j , usamos função

$step(\mathbf{C}_j)$, onde \mathbf{C}_j é j -ésimo valor em \mathbf{C} . A função $step$ é, novamente, dada por:

$$step(y) = \begin{cases} 1 & \text{se } y \geq 0 \\ -1 & \text{caso contrário.} \end{cases}$$

Depois de calcular \mathbf{C} , atualizamos o vetor \mathbf{w} de pesos da seguinte maneira:

$$\mathbf{w} = \mathbf{w} - \alpha \frac{1}{M} \mathbf{X}(\mathbf{C} - \mathbf{Z})$$

Aqui, \mathbf{X} é a matriz de exemplos, α é a *taxa de aprendizado*, \mathbf{Z} é a matriz com as classificações corretas de cada exemplo em \mathbf{X} e \mathbf{C} é a matriz de predições feita pelo algoritmo nessa iteração.

A minha implementação do *Gradient Descent* executa esse procedimento até que o número de iterações seja maior do que um limite ρ . Nesse *dataset*, utilizei $\rho = 500$. Depois do treinamento do modelo, obtemos um vetor \mathbf{w}^* , que é usado para gerar as predições para cada *tweet* contido nos dados para teste.

4 Conclusão

A minha implementação do *Perceptron* teve um desempenho melhor do que a implementação do *Gradient Descent* mas, usando metade dos exemplos como treinamento e outra metade como teste, ambos os algoritmos atingiram resultados de por volta de 10% de exemplos classificados erroneamente.