Aprendizado de Máquina

Aula 12

http://www.ic.uff.br/~bianca/aa/

Tópicos

1. Introdução – Cap. 1 (16/03)
2. Classificação Indutiva – Cap. 2 (23/03)
3. Árvores de Decisão – Cap. 3 (30/03)
4. Ensembles - Artigo (13/04)
5. Avaliação Experimental – Cap. 5 (20/04)
6. Aprendizado de Regras – Cap. 10 (27/04)
7. Redes Neurais – Cap. 4 (04/05)
8. Teoria do Aprendizado – Cap. 7 (11/05)
9. Máquinas de Vetor de Suporte – Artigo (18/05)
10. Aprendizado Bayesiano – Cap. 6 e novo cap. online (25/05)
11. Aprendizado Baseado em Instâncias – Cap. 8 (01/05)
12. **Classificação de Textos – Artigo (08/06)**
13. Aprendizado por Reforço – Artigo (15/06)

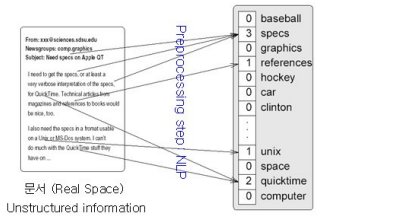
Aplicações de Classificação de Textos

- Páginas web
  - Recomendação
  - Classificação em tópicos (ex.: hierarquia do Yahoo)
- Mensagens de fóruns/blogs
  - Recomendação
  - Filtragem de spam
  - Análise de sentimentos (em relação a produtos)
- Artigos de jornal
  - Personalização
- Mensagens de e-mail
  - Priorização
  - Separação em pastas
  - Filtragem de spam
  - Colocação de anúncios (Gmail)

Representação de Textos

- Modelo mais comum é o *Bag-of-Words*
  - A ordem em que as palavras aparecem é desconsiderada
  - Um atributo por palavra, podendo ser
    - Booleano = indica a presença da palavra
    - Numérico = indica a frequência
  - Palavras sem significado (chamadas de stopwords) são removidas.
    - Ex.: artigos, pronomes

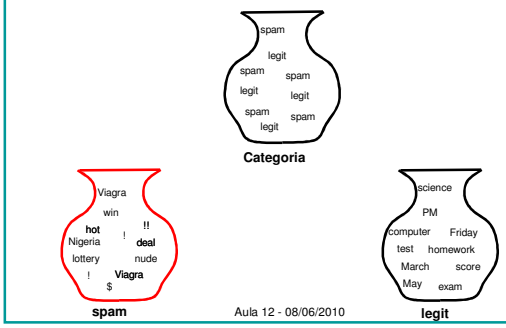
Modelo *Bag-of-Words*



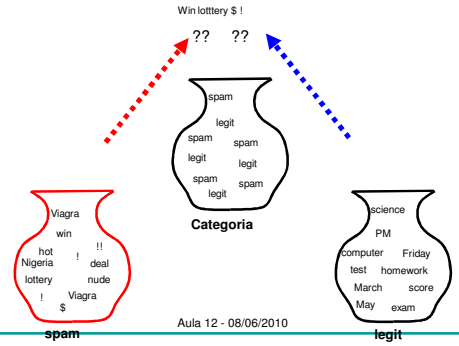
Métodos de Classificação de Textos

- Representações de texto tem alta dimensão.
  - Um atributo por palavra.
- Vetores são esparsos porque muitas palavras são raras.
  - Lei de Zipf
- Algoritmos com alto viés que previnem super-ajuste em altas dimensões são os melhores.
- Para a maioria dos problemas de classificação de textos, há muitos atributos relevantes.
- Métodos que somam evidências de muitos atributos (como naïve Bayes, KNN, rede neural, SVM) funcionam melhor do que os que isolam alguns atributos relevantes (árvore de decisão ou indução de regras).

Modelo Naïve Bayes para textos



Classificação Naïve Bayes



Algoritmo Naive Bayes para Textos (Treinamento)

Seja D um conjunto de documentos
Seja V o vocabulário de todas as palavras nos documentos de D
Para cada classe $c_i \in C$
  Seja D_i o subconjuntos de documentos em D que pertencem à categoria c_i
  $P(c_i) = |D_i| / |D|$
  Seja T_i a concatenação de todos os documentos em D_i
  Seja n_i o número total de ocorrências de palavras em T_i
  Para cada palavra $w_j \in V$
    Seja n_{ij} o número de ocorrências de w_j em T_i
    Let $P(w_{ij} | c_i) = (n_{ij} + 1) / (n_i + |V|)$

Algoritmo Naive Bayes para Textos (Teste)

Dado um documento de teste X
Seja n o número de ocorrências de palavras em X
Retorne a classe:

$$\underset{c_i \in C}{\operatorname{argmax}} P(c_i) \prod_{i=1}^{n} P(a_i \mid c_i)$$

onde a_i é a palavra que ocorre na i-ésima posição de X

Prevenção de *Underflow*

- Multiplicar muitas probabilidades, que estão entre 0 e 1, pode resultar num *underflow* de ponto flutuante.
- Como $\log(xy) = \log(x) + \log(y)$, é melhor fazer todos os cálculos somando logs de probabilidades ao invés de multiplicar probabilidades.
- Classe com maior valor de log-probabilidade é também a mais provável na escala normal.

Métricas de Similaridade de Texto

- Medir a similaridade de textos é um problema bastante estudado.
- Métricas são baseadas no modelo "*bag of words*".
- Normalmente é feito um pré-processamento: "*stop words*" são removidas e as palavras são reduzidas à sua raiz morfológica.
- Modelo vetorial de Recuperação de Informação (IR) é a abordagem padrão.

O modelo vetorial

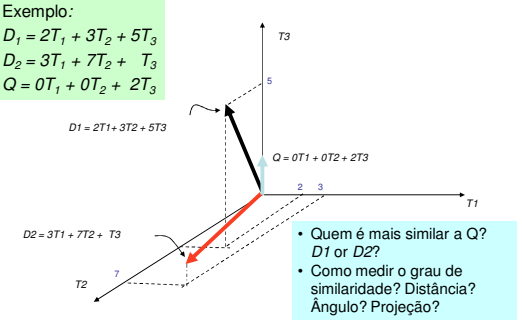
- Supõe-se que *t* termos distintos restam após o pré-processamento; chamados de termos do vocabulário.
- Estes termos "ortogonais" formam um espaço vetorial.

  Dimensão = *t* = |vocabulário|
- Cada termo, *i*, num documento ou consulta, *j*, tem um peso dado por um número real, w_{ij}.
- Tanto documentos quando consultas são representados por vetores t-dimensionais:

$$d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$$

Representação gráfica

Exemplo:
$D_1 = 2T_1 + 3T_2 + 5T_3$
$D_2 = 3T_1 + 7T_2 + T_3$
$Q = 0T_1 + 0T_2 + 2T_3$



- Quem é mais similar a Q? *D1* or *D2*?
- Como medir o grau de similaridade? Distância? Ângulo? Projeção?

Coleção de Documentos

- Uma coleção de *n* documentos pode ser representada no modelo vetorial por uma matriz.
- Uma entrada na matriz corresponde ao "peso" do termo no documento; zero indica que o termo não é significativo no documento ou simplesmente não existe no documento.

	T_1	T_2	....	T_t
D_1	w_{11}	w_{21}	…	w_{t1}
D_2	w_{12}	w_{22}	…	w_{t2}
:	:	:		:
:	:	:		:
D_n	w_{1n}	w_{2n}	…	w_{tn}

Pesos: Frequência dos Termos

- Termos frequentes em um documento são mais importantes, i.e. mais indicativos do tópico do documento.

  f_{ij} = frequência do termo *i* no documento *j*
- Podemos obter a *frequência do termo* (*tf*) dividindo *f* pela frequência do termo mais comum no documento:

$$tf_{ij} = f_{ij} / max_i\{f_{ij}\}$$

Pesos: Frequência Inversa dos Documentos

- Termos que aparecem em muitos documentos *diferentes* são *menos* significativos.

  df_i = frequência em documentos do termo *i*

  = número de documentos contendo o termo *i*

  idf_i = frequência inversa em documentos do termo *i*,

  $= \log_2 (N/ df_i)$

  (N: número total de documentos)
- É uma indicação do *poder de discriminação* do termo.
- Log é usado para diminuir o efeito em relação a *tf*.

Ponderação TF-IDF

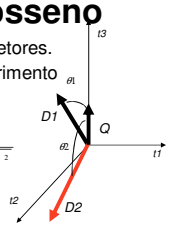
- Uma ponderação tipicamente utilizada é:

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N/ df_i)$$
- Um termo que ocorre com frequência no documento mas raramente no resto da coleção tem peso maior.
- Muitas outras formas de ponderação foram propostas.
- Experimentalmente, determinou-se que a ponderação *tf-idf* funciona bem.

Medida de Similaridade de Cosseno

- Mede o cosseno do ângulo entre dois vetores.
- Produto interno normalizado pelo comprimento dos vetores.

$$\text{CosSim}(\mathbf{d_j}, \mathbf{q}) = \frac{\vec{d}_j \cdot \vec{q}}{\left|\vec{d}_j\right| \cdot \left|\vec{q}\right|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$



D1 = 2T1 + 3T2 + 5T3 CosSim(*D1* , *Q*) = 10 / √(4+9+25)(0+0+4) = 0.81
D2 = 3T1 + 7T2 + 1T3 CosSim(*D2* , *Q*) = 2 / √(9+49+1)(0+0+4) = 0.13
Q = 0T1 + 0T2 + 2T3

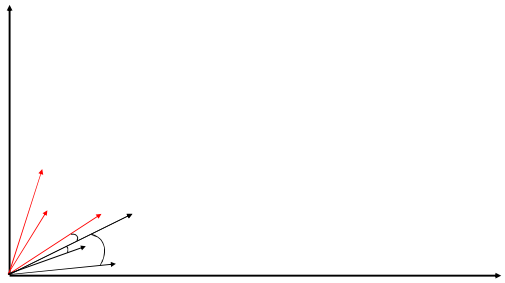
D1 é 6 vezes melhor que *D2* usando similaridade de cosseno mas só 5 vezes melhor usando produto interno.

K-NN para Textos

Treinamento:
Para cada exemplo de treinamento <*x*, *c*(*x*)> ∈ *D*
Calcule o vetor TF-IDF correspondente, $\mathbf{d}_x$, para o documento *x*

Exemplo de teste *y*:
Calcule o vetor TF-IDF **d** para o documento *y*
Para cada <*x*, *c*(*x*)> ∈ *D*
Seja s_x = cosSim(**d**, $\mathbf{d}_x$)
Ordene os exemplos, *x*, em *D* por valor decrescente de s_x
Seja *N* o conjunto dos primeiros *k* exemplos de D.
Retorne a classe majoritária dos exemplos em *N*.

Exemplo: 3-NN para Textos



Índice Invertido

- Busca linear na base de treinamento não é escalável.
- Índice invertido: estrutura mapeando palavras a documentos.
- Quando as *stopwords* são removidas, as palavras que sobram são raras, então um índice invertido ajuda a eliminar boa parte dos documentos que não tem muitas palavras em comum com o documento de teste.

Conclusões

- Existem muitas aplicações importantes da classificação de textos.
- Requer uma técnica que lide bem com vetores esparsos de muitos atributos, porque tipicamente cada palavra é um atributo e a maioria das palavras é rara.
  - Naïve Bayes
  - kNN com similaridade de cosseno
  - SVMs