**Prática de Laboratório 2: Algoritmos Naïve Bayes e kNN**

Nesta prática serão utilizados dois métodos para treino de classificadores para conjuntos de dados concretos. Também se examinará um procedimento para a seleccção do valor ideal do parâmetro associado a um algoritmo ao treinar um classificador e outro para a selecção de um subconjunto de atributos, o mais efectivo, para a construção de um classificador.

Na mesma serão utilizados os conjuntos de dados *Spambase* e *Glass*.

Com a prática se pretende alcançar os seguintes objectivos:

* Utilizar os métodos Naïve Bayes e kNN para treinar classificadores para conjuntos de dados concretos
* Analisar o modelo aprendido ao treinar um classificador Naïve Bayes
* Analisar uma estratégia de selecção do valor ideal para o parâmetro de um classificador com um conjunto de dados concreto
* Explorar o efeito produzido no rendimento de um classificador kNN pela introdução de ruído num conjunto de dados
* Analisar uma estratégia de selecção do subconjunto de atributos mais efectivo para a criação de um classificador

Como pré-requisitos é importante o estudo de diversos aspectos:

* Avaliação dos modelos e métodos de validação
  + Medidas de desempenho, *holdout*, validação cruzada, matriz de confusão
  + Consultar: Witten, pgs. 147 – 150; 152 – 157; 180 - 182
* Algoritmos Naïve Bayes e KNN
  + Consultar: Mitchell, pg. 154 – 158, 177 – 184, 230 – 236
  + Selecção de atributos
  + Consultar: Witten, secção 7.1, pg. 307 – 314

**Tarefa 1: filtragem de *spam***

Nesta actividade utilizaremos o Weka para treinar um classificador Naïve Bayes, para efeitos de detecção de *spam*.

Como conjunto de dados utilizaremos o conjunto *Spambase*, constituído por um conjunto de *e-mails* marcados a partir de uma única conta de *e-mail*.

Leia a descrição do conjunto de dados disponível em <https://archive.ics.uci.edu/ml/datasets/Spambase> para ter uma ideia acerca do mesmo. Os dados foram convertidos para o formato ARFF (ficheiro *spambase.arff* ).

Algumas operações simples de pré-processamento serão necessárias antes dos dados estarem prontos a ser utilizados. Podemos fazer isso com o Weka:

1. No painel *Preprocess* carregue o conjunto de dados *spambase.arff*.
2. A lista completa dos atributos do conjunto de dados aparecerá na secção "*Attributes*".
3. Elimine os atributos *capital\_run\_length\_average*, *capital\_run\_length\_longest* e *capital\_run\_length\_total*, marcando-os na caixa à sua esquerda e fazendo *click* no botão *Remove*.
4. Os demais atributos representam frequências relativas de várias palavras e caracteres importantes presentes em mensagens de *e-mail*. Queremos converter as referidas frequências em valores booleanos: 1 se a palavra ou caracter está presente no *e-mail*, 0 se não está presente. Para tal, selecione o botão *Choose* na secção *Filter* na parte superior da janela e escolha *Filters > unsupervised > attribute > NumericToBinary* e *click* no botão *Apply*. Todas as frequências numéricas nos atributos são convertidos para valores booleanos. Cada *e-mail* é agora representado por um vector com 55 dimensões que representa se uma determinada palavra existe ou não no *e-mail*. Esta forma de representação é denominada *bag of words* (é uma forma de representação simplista uma vez que não tem em conta a ordem das palavras).
5. Guarde os dados pré-processados para posterior uso. Utilize para tal o botão *Save*.

Com os dados carregados, desejamos treinar um classificador Naïve Bayes para diferenciar os *e-mails* regulares do *spam* ajustando-se à distribuição do número de ocorrências de cada palavra nas duas classes de *e-mail*. No separador *Classify*:

1. Faça *click* no botão *Choose* e seleccione *classifiers > Bayes > Naivebayes*.
2. Com as opções seleccionadas por defeito, faça *click* no botão *Start* para construir o classificador. Analise a saída produzida, particularmente as percentagens de instâncias classificadas correcta e incorrectamente.
   1. Quanto tempo empregou o classificador ao treinar e classificar os dados?
3. Examine o modelo produzido pelo Weka (apresentado antes da informação sobre o rendimento do classificador). Observe as probabilidades *a priori* para cada classe.
   1. Como calcula o algoritmo a probabilidade de um *e-mail* pertencer a uma classe (*spam* ou *Não spam*)?
   2. Determine a probabilidade condicional de observar a palavra "3d" dado que uma mensagem pertence à classe *spam* P(3d|*spam*) ou *não spam* P(3d|*não spam*). Para isso utilize as contagens produzidas pelo modelo, apresentadas na janela de saída (*Classifier output*) no separador *Classify*.
4. Assumamos agora que somos *spammers* tentando burlar um sistema de detecção de *spam* baseado num classificador Naïve Bayes no sentido de classificar uma mensagem *spam* como *ham* (i.e. uma mensagem válida). Utilizemos o conjunto de dados de treino completo para treinar o classificador e apliquemos o modelo aprendido a um conjunto de teste dedicado. Carregue o conjunto de teste no Weka. No separador *Classify*, seleccione *supplied test set > set > open file* e seleccione o ficheiro *spambase\_test.arff*. Este ficheiro contém um vector de dados binário representando um *e-mail* considerado *spam*. Execute o classificador sobre este conjunto de teste. É o *e-mail* classificado correctamente?
5. Abra o ficheiro de teste *spambase\_test.arff* num editor de texto (por ex. o bloco de notas). Identifique boas palavras, características da classe *não spam* e acrescente-as ao *e-mail*. Importante: não altere a etiqueta de classe do ficheiro de teste (último valor dos atributos). Ao testar o Weka ignora este atributo e utiliza o classificador treinado previamente para predizer a etiqueta correspondente ao *e-mail*. Execute novamente o classificador com o ficheiro de teste modificado. Alterou-se a classe atribuída ao *e-mail*?

**Tarefa 2: classificação de vidro**

Nesta actividade vamos experimentar a classificação com o método de *k vizinhos mais próximos*. Na mesma utilizaremos o conjunto de dados *Glass*. Este conjunto foi criado pelo Serviço de Ciência Forense dos Estados Unidos (*U. S. Forensic Science Service*) e contém dados acerca de seis tipos diferentes de vidro. O vidro é descrito pelo seu índice de refracção e pelos elementos químicos contidos no mesmo. O conjunto está disponível no repositório de dados da UCI

(<https://archive.ics.uci.edu/ml/datasets/Glass+Identification>).

1. No Explorador do Weka, carregue o ficheiro *glass.arrf*.
   1. Quantos atributos tem o conjunto de dados?
   2. Quais são os seus nomes?
   3. Qual é o atributo correspondente à classe?
2. No separador *Classify*, seleccione *classifiers > lazy > IBk*. Esta opção corresponde ao método *dos k vizinhos mais próximos* (kNN).
3. No painel *Test options*, seleccione *Use training set* e pressione o botão *start*.
4. Observe a saída do classificador e considere o seguinte:
   1. Qual é o rendimento do classificador?
   2. Porquê se considera uma má ideia testar um classificador kNN (k = 1) sobre o próprio conjunto de treino?
   3. Se espera que a classificação sobre um conjunto de teste seja tão boa quanto esta?
5. Agora vamos avaliar o classificador utilizando uma validação cruzada. Execute o classificador, utilizando o número de folhas fixado por defeito (10).
   1. Como é o rendimento do classificador relativamente ao caso anterior?
6. Exploremos agora o efeito do parâmetro k. Para tal, execute várias vezes o classificador com diferentes valores do parâmetro (1, 3, 5, 10, 15), utilizando sempre para a avaliação a validação cruzada em 10 folhas. Recorde que se pode alterar o parâmetro do algoritmo no painel de opções, fazendo *click* diante do botão *Choose*.
   1. Como é afectado o rendimento do classificador com a variação do parâmetro k?
   2. Represente graficamente os resultados (valor de k no eixo x e o rendimento no eixo y). É possível concluir algo observando o gráfico?

O método dos vizinhos mais próximos, a semelhança de outros métodos de aprendizagem, é sensível à presença de ruído nos dados de treino. Agora faremos a injecção de certa percentagem de ruído nos valores de classe e observaremos o seu efeito no rendimento do classificador. O ruído pode ser introduzido através do filtro *AddNoise* (*filters > unsupervised > attribute > AddNoise*). Neste caso é importante que o ruído seja introduzido somente nos dados de treino, permanecendo inalterados os dados de teste. Para tal utilizaremos um *metaclassificador*, designado *FilteredClassifier* (*classifiers > meta > FilteredClassifier*). O *metaclassificador* deve ser configurado para utilizar *IBk* como classificador e *AddNoise* como filtro. Isto pode ser efectuado utilizando a correspondente janela de opções.

1. Construa uma tabela com o rendimento estimado do classificador, através da validação cruzada em 10 folhas, para seis diferentes percentagens de ruído (0%, 10%, ..., 50%) e diferentes valores de k (1, 3 e 5).
   1. Que efeito provoca o aumento do ruído?
   2. Qual é efeito derivado da alteração do valor de k?

**Tarefa 3: selecção de atributos**

Investiguemos agora que subconjunto dos atributos produz o menor erro de validação cruzada sobre o conjunto de dados *Glass*, utilizando o algoritmo *KNN*.

A realização de uma busca exaustiva de todos os subconjuntos possíveis é impraticável (porquê?), por isso empregaremos um procedimento de eliminação para trás (ver ref. Witten). Para tal, primeiro consideremos a eliminação de cada atributo individualmente do conjunto de dados completo e a realização de uma validação cruzada para cada versão reduzida do conjunto de dados. Uma vez determinado o melhor subconjunto de oito atributos, repete-se o procedimento para a escolha do melhor subconjunto de sete atributos e assim sucessivamente.

1. Construa uma tabela mostrando o melhor subconjunto de atributos e a menor taxa de erro obtidos em cada iteração. Compare as taxas de erro obtidas neste processo com a correspondente ao conjunto de dados completo.

| **Tamanho do Subconjunto**  **(Nº de Atributos)** | **Atributos no “Melhor” Subconjunto** | **Taxa de Erro** |
| --- | --- | --- |
| 9 |  |  |
| 8 |  |  |
| 7 |  |  |
| 6 |  |  |
| 5 |  |  |
| 4 |  |  |
| 3 |  |  |
| 2 |  |  |
| 1 |  |  |