EACH | campus capital USP LESTE
Escola de Artes, Ciências e Humanidades
Universidade de São Paulo

Atividade Final

SIN5007- Reconhecimento de Padrões

Profa. Ariane Machado Lima

Novembro, 2017

Integrantes

- Lara Marinelli - 7971938
    - (Orientadora: Patrícia Rufino em parceria com Anna Karenina Azevedo Martins)

- Maria Sinesia Matias - 10543335
    - Aluna Especial

1. Conjunto de Dados

O conjunto de dados (retirado da competição kaggle) é referente aos sinais de proteínas (Histonas) em genes. A idéia é partir dos sinais dessas proteínas, predizer se o gene está mais expresso ou menos expresso.

- **5** características **discretas** além do rótulo e id;
- **1.548.500** instâncias no conjunto sem pré processamento (onde cada 100 instâncias equivalem a um gene [matriz 100x5];
- *Obs: após o pré-processamento, o conjunto passou a ter **15485** instâncias.*
- 2 classes: Alto nível de expressão (1) e Baixo nível de expressão (0);
- Não há instâncias com missing values.

2- É possível usar PCA?

Foi avaliada a covariância entre as variáveis do conjunto utilizando o comando **cov (R) .** Os resultados mostraram dependência entre os dados (todos os valores distantes de zero):

```
> cov(newdata)
            H3K4me3   H3K4me1  H3K36me3    H3K9me3 H3K27me3
H3K4me3   18400.900 -1650.233 -7018.161 -20143.018 2197.141
H3K4me1   -1650.233  7338.201 11922.509  23358.956 2240.037
H3K36me3  -7018.161 11922.509 62244.922  85960.492 2318.138
H3K9me3  -20143.018 23358.956 85960.492 231121.356 5271.439
H3K27me3   2197.141  2240.037  2318.138   5271.439 3835.228
```

Figura 1: Matriz de covariância do conjunto

2- É possível usar PCA?

É possível ver a dependência entre as variáveis analisando-se a matriz de correlação **(cor(R))**. O que significa que **o PCA pode ser aplicado.**

```
> cor(newdata)
             H3K4me3    H3K4me1   H3K36me3    H3K9me3  H3K27me3
H3K4me3    1.0000000 -0.1420140 -0.2073728 -0.3088766 0.2615428
H3K4me1   -0.1420140  1.0000000  0.5578545  0.5672031 0.4222453
H3K36me3  -0.2073728  0.5578545  1.0000000  0.7166826 0.1500347
H3K9me3   -0.3088766  0.5672031  0.7166826  1.0000000 0.1770572
H3K27me3   0.2615428  0.4222453  0.1500347  0.1770572 1.0000000
```

Figura 2: Matriz de correlação do conjunto

3. PCA

Para a realização do PCA foi utilizada a função **prcomp,** do **R.** Segundo Landeiro (2011), há duas maneiras de se fazer o cálculo dos componentes principais, utilizando a **covariância** e a **correlação**. Nós optamos por testar e comparar os dois métodos. Segundo o autor, o uso da função prcomp sem parâmetros adicionais além do dataset permite o cálculo por covariância **(prcomp(newdata), no nosso caso)**. O cálculo por correlação é obtido passando-se o parâmetro **Scale=TRUE.**

4. PCA utilizando covariância - Resultados

- Com base nos resultados, e de acordo com Jolliffe(2002) optamos por usar os **três primeiros componentes** (proporção acumulada de 0.97617) pois os últimos componentes não agregam muita informação ao conjunto.

```
> #proporcao dos componentes
> summary(resu.pca)
Importance of components%s:
                              PC1       PC2       PC3      PC4     PC5
Standard deviation     521.2557 163.00014 130.26734 74.35895 46.5321
Proportion of Variance   0.8414   0.08227   0.05255  0.01712  0.0067
Cumulative Proportion    0.8414   0.92363   0.97617  0.99330  1.0000
```

Figura 3: Resultado do PCA de covariância sumarizado

5. PCA utilizando correlação - Resultados

- Já no PCA utilizando correlação optamos por escolher **os quatro primeiros componentes** (proporção acumulada de 0.94633) pois o quinto componente não agrega muita informação ao conjunto.

```
> summary(resu.pcacorr)
Importance of components%s:
                          PC1    PC2    PC3     PC4     PC5
Standard deviation     1.5575 1.1484 0.7828 0.61172 0.51801
Proportion of Variance 0.4852 0.2638 0.1226 0.07484 0.05367
Cumulative Proportion  0.4852 0.7489 0.8715 0.94633 1.00000
```

Figura 4: Resultado do PCA de correlação sumarizado

6. Relief

Para a seleção de características, foi utilizado o **Relief (pacote FSelector do R).** Valores dos parâmetros:

- **neighbours.count = 10 (**valor recomendado na documentação)
- **sample.size = 20** (ajustado de acordo com o tempo disponível)
- **Cutoff = 3**

Características selecionadas: "H3K9me3" "H3K4me1" "H3K27me3”

7. Método de avaliação

Como forma de avaliação, foi utilizado o K-fold cross-validation, com k=5. A distribuição das classes ficou equilibrada:

	Classe 0	Classe 1
Dados totais	7757	7728
Fold 1	1551	1545
Fold 2	1551	1545
Fold 3	1551	1545
Fold 4	1551	1545
Fold 5	1551	1545

Figura 6: Distribuição das classes das folds

8. Métricas de avaliação

As medidas comparadas serão os valores médios de precisão, sensibilidade e erro total .

Precisão: TP/(TP+FP) (significa, dos itens que foram classificados como positivos, quantos realmente eram positivos)

Sensibilidade (ou recall): TP/(TP+FN) (significa, dos itens que eram positivos, quanto o classificador acerta)

Erro total: (FP+FN)/(TP+TN) (significa o quanto que o classificador errou)

9. SVM

Foi utilizada a função svm {e1071}, do R.

A função foi invocada na seguinte forma:

Call:

svm(formula = Prediction ~ ., data = dadostreinamento, type = "C-classification")

Os dados foram variados para análises.

10. Redes Neurais

Foi utilizado o pacote nnet (R). Parâmetros:
Decay (taxa de aprendizado)
Size (número de neurônios da camada oculta)

Executando podemos perceber que a rede convergiu rápido quando os pesos foram iniciados próximos a 0 e com mais neurônios.
O tempo aumentou na execução quando os pesos foram iniciados próximos a 1 e menos neurônios.
Comparado aos outros métodos, executou rápido como o NaveBayes.
Porém foi confuso interpretar os resultados obtidos nas execuções.

10. Redes Neurais

Isso quanto a forma de exibição dos resultados na ferramenta.
Resultado da execução:

```
> result_nnet1 <- nnet(Prediction~., fold, rang=0.1,size=5, maxit=500, subset=NULL, Hess=TRUE, na.action=NULL, MaxNWts=1000,contrasts=
# weights:  41
initial  value 796.222054
final  value 498.000000
converged
> result_nnet1 <- nnet(Prediction~., fold, rang=0.8,size=5, maxit=500, subset=NULL, Hess=TRUE, na.action=NULL, MaxNWts=1000,contrasts=
# weights:  41
initial  value 795.703900
iter  10 value 389.085333
final  value 388.658269
converged
> result_nnet1 <- nnet(Prediction~., fold, rang=0.8,size=2, maxit=500, subset=NULL, Hess=TRUE, na.action=NULL, MaxNWts=1000,contrasts=
# weights:  17
initial  value 775.824242
iter  10 value 367.218229
iter  20 value 346.859186
iter  30 value 330.161706
iter  40 value 328.631588
iter  50 value 295.185783
iter  60 value 287.814656
iter  70 value 287.479397
final  value 287.479224
converged
```

10. Redes Neurais

Resultado da execução:

result_nnet1	list [19] (S3: nnet.formula, nnet)	List of length 19
n	double [3]	6 5 1
nunits	integer [1]	13
nconn	double [14]	0 0 0 0 0 0 ...
conn	double [41]	0 1 2 3 4 5 ...
nsunits	integer [1]	13
decay	double [1]	0
entropy	logical	FALSE
softmax	logical	FALSE
censored	logical	FALSE
value	double [1]	357.7889
wts	double [41]	-0.0947 -0.0638 0.0775 -0.1151 -0.2308 -0.0564 ...
convergence	integer [1]	0
fitted.values	double [3096 x 1]	0.139 0.139 0.873 0.873 0.873 0.139 ...
residuals	double [3096 x 1]	-0.139 -0.139 0.127 0.127 -0.873 -0.139 ...
call	language	nnet.formula(formula = Prediction ~ ., data = fold, rang = 0.1, size = 5, m ...
Hessian	double [41 x 41]	-6.81e-09 -1.36e-08 -8.78e-07 -3.40e-07 -2.38e-07 -3.47e-07 -1.36e-08 -2.72e-08 ...
terms	formula	Prediction ~ GeneId + H3K4me3 + H3K4me1 + H3K36me3 + H3K9me3 + H3K27me3
coefnames	character [6]	'GeneId' 'H3K4me3' 'H3K4me1' 'H3K36me3' 'H3K9me3' 'H3K27me3'
xlevels	list [0]	List of length 0

Redes Neurais

result	0	1
0	904	102
9.15779970883928e-07	1	0
0.0114068420550077	1	0
0.0743894146832188	1	0
0.3136232164118	1	0
0.725437559522833	1	0
0.739972585753912	1	0
0.740280247490663	1	0
0.747330462255735	1	0
0.747598923240564	1	0
0.747694847426298	1	0
0.747708366834617	1	0
0.747757859945452	1	0
0.747891072101699	1	0
0.747947367988373	1	0
0.74795908818309	1	0
0.747962222134725	0	1
0.747966842944924	626	1448

11. Naive Bayes

Foi utilizado o pacote naivebayes, do R.
Parâmetros utilizados:
prior = NULL
laplace = 0
usekernel = FALSE

Chamada para criar o modelo:
model<- naive_bayes(Prediction ~ ., data = dados)

Discussão

O PCA de correlação apresentou menor retenção de variabilidade do conjunto nos primeiros componentes.Mais componentes, detendo menor variabilidade dos dados. No entanto, nos classificadores SVM e Naive Bayes, os dados selecionados utilizando-o apresentaram os melhores resultados.
Selecionamos três características utilizando o Relief, e seu desempenho nos classificadores foi muito pior em relação ao PCA e ao conjunto todo.

Discussão

No geral, não tivemos grandes variações nos resultados com a variação dos parâmetros com o kernel linear. Já com o kernel polinomial, sigmoide e radial, custo e erro aumentaram juntos. Isso se dá pelo fato de que um aumento do custo pode aumentar o overfitting.

Ao contrário dos demais, no Kernel sigmoide, o conjunto todo apresentou melhores resultados do que os com redução de dimensionalidade.

No geral, dentro de configurações semelhantes, se aumentarmos o **gamma,** aumentamos o erro. Isso se dá pelo fato de que aumentando o valor do gamma, diminuímos a influência de pontos distantes na definição da divisão.

Resultados

O desempenho do classificador Naive Bayes foi péssimo em relação aos demais. Imaginamos que isso esteja relacionado com uma possível dependência dos atributos dada a classe, contrariando a suposição na qual o NB se baseia.

	Erro	Sensibilidade	Precisão
Todas as Características	0.4990310078	0	0
Relief	0.4990310078	0	0
PCA Corr	0.4937984496	0.5258514833	0.1038187702
PCA Cov	0.4989664083	0	0.00012944983

Conclusão

Após testar os classificadores SVM e Naive Bayes, (a parte os problemas que tivemos com os resultados das Redes neurais), o melhor classificador obtido foi:

Características	Kernel	Gamma	Coef0	Degree	Custo	Erro Total	Sensibilidade	Precisão
PCA corr	Linear	0.142857142	-	-	1	0.15168	0.860	0.835

Apêndice

Todos os códigos utilizados para fazer as avaliações estão compilados no arquivo: **Codigos_Reconhecimento_de_Padroes.pdf**

Referências

- BOSWELL, D. Introduction to Support Vector Machines. 2002
- Documentação do R: svm Disponível em: http://127.0.0.1:23403/library/e1071/html/svm.html
- Desafio: https://inclass.kaggle.com/c/gene-expression-prediction
- LANDEIRO, V. L. Introdução ao uso do programa R. Instituto Nacional de Pesquisas da Amazônia. Programa de Pós Graduação em Ecologia, 2011. Disponível em: https://cran.r-project.org/doc/contrib/Landeiro-Introducao.pdf
- JOLLIFFE, I. T. Principal Component Analysis. 2. ed. Springer, 2002 (Cap. 6)

Referências

Pacote “FSelector”

https://cran.r-project.org/web/packages/FSelector/FSelector.pdf