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

Classificação Binária e Aprendizado Supervisionado

SIN5007- Reconhecimento de Padrões

Profa. Ariane Machado Lima

Novembro, 2017

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

1. Definição do Conjunto de Dados

SIN5007- Reconhecimento de Padrões
Profa. Ariane Machado Lima

Agosto, 2017

Integrantes

- Lara Marinelli - 7971938
    - (Orientadora: Patrícia Rufino em parceria com Anna Karenina Azevedo Martins)
- Maria Sinesia Matias - 10543335
    - Aluna Especial

Sobre a Escolha do Conjunto de Dados

O trabalho consiste em aplicar técnicas de aprendizado supervisionado para realizar classificação binária.

Desta forma, foi escolhido um conjunto de dados rotulados, com duas classes.

Sobre o Conjunto de Dados

- Gene Expression Prediction
- Proveniente de uma competição (finalizada em 05/03/17) do Kaggle intitulada: Predicting gene expression from histone modification signals
- Em biologia, as histonas são as principais proteínas que compõem o nucleossomo e tem um papel importante na regulação dos genes
- O desafio é predizer o nível de expressão do gene (Alto nível de expressão=1, baixo nível de expressão=0) baseado nos marcadores das histonas.

Descrição do Conjunto de Dados

- The dataset of this competition is on "E047" (Primary T CD8+ naive cells from peripheral blood) celltype from Roadmap Epigenomics Mapping Consortium (REMC) database. For each gene, it has 100 bins with five core histone modification marks. (We divide the 10,000 basepair (bp) DNA region (+/-5000bp) around the transcription start site (TSS) of each gene into bins of length 100 bp, and then count the reads of 100 bp in each bin. Finally, the signal of each gene has a shape of 100x5.)

Descrição do Conjunto de Dados (português)

- -O conjunto de dados desta competição está no tipo de célula "E047" (Primary T CD8 + naive cells from peripheral blood) do banco de dados Epigenomics Mapping Consortium (REMC). Para cada gene, o conjunto possui 100 instâncias com cinco marcas de modificação de histonas principais [1]. (Os autores dividiram a região de DNA de 10.000 pares de bases (pb) (+/- 5000bp) em torno do início da transcrição (TSS) de cada gene em instâncias de comprimento de 100 pb [2] e, em seguida, conta as leituras de 100 pb em cada lixeira. Finalmente, o sinal de cada gene tem uma forma de 100x5.)

Informações do Conjunto de Dados

- 5 características além do rótulo e id;
- 1.548.500 instâncias (onde cada 100 instâncias equivalem a um gene [matriz 100x5];
- 2 classes: alto nível de expressão (1) e baixo nível de expressão (0);
- Não há instâncias com missing values.

Arquivos

Conjunto de treinamento contém dois arquivos:

Arquivo x_train: *contém os dados sem o rótulo e com as características*

Arquivo y_train: *contém a classificação/rótulo dos dados*

Atributos do Arquivo X-Train

GeneId	Identificador do gene	
H3K4me1	Marcador de modificação da histona	Variável discreta
H3K36me3	Marcador de modificação da histona	Variável discreta
H3K27me3	Marcador de modificação da histona	Variável discreta
H3K9me3	Marcador de modificação da histona	Variável discreta
H3K4me3	Marcador de modificação da histona	Variável discreta

Atributos do Arquivo Y-Train

GeneId	Identificador do gene	
Prediction	Classificação do gene	Variável nominal

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

2. Redução de Dimensionalidade Utilizando PCA

SIN5007- Reconhecimento de Padrões

Profa. Ariane Machado Lima

Setembro, 2017

Introdução Sobre PCA

PCA (*principal component analysis*) é um método de redução de dimensionalidade (características), mais indicado quando o conjunto de dados possui mais características do que dados, e estas características tem dependência (covariância) e alta correlação.

O PCA quando aplicado, transforma as características em novas (fusão), buscando uma quantidade ótima para o conjunto de dados. Estas novas características (componentes) não são dependentes, e recomenda-se escolher os componentes com maior variância para separar as classes.

O objetivo do PCA é minimizar problemas relacionados a maldição da dimensionalidade.

Objetivo

O conjunto de dados usado neste trabalho, não possui uma quantidade maior de características em relação aos dados, minimizando as chances de ocorrer a maldição da dimensionalidade.

Será aplicado o PCA no conjunto de dados, a fim de analisar os componentes e reduzir a dimensionalidade, e avaliar os resultados:

- quais e quantos componentes tem maior variabilidade;
- impactos no classificador usando todos os componentes vs componentes selecionados aplicando PCA.

Métodos – Pré Processamento

O pré processamento realizado no conjunto de dados, consistiu em agrupar as informações dos genes, usando o pacote *dplyr* em R.

No arquivo x-train, as informações dos genes estão distribuídas em uma matriz 100x5. Ou seja, 100 instâncias deste arquivo equivalem a 01 gene.

A fim de otimizar a aplicação das técnicas abordadas neste trabalho ao conjunto de dados, foi feito um somatório em cada característica e para cada gene (matriz 100x5).

Métodos – Pré Processamento

Antes: 1.548.500 instâncias, onde a cada 100 equivale a 01 gene [100x5]

Depois: 15.485 instâncias, onde cada instância equivale a 01 gene [1x5]

```
#Pre processamento
install.packages("dplyr")

x_train<-read.csv(file=".../x_train.csv",header=T,sep=",")

library(dplyr)

newdata<-group_by(x_train,GeneId)%>%summarise(H3K4me3=sum(H3K4me3),H3K4me1=sum(H3K4me1),H3K36me3=sum(H3K36me3),
H3K9me3=sum(H3K9me3),H3K27me3=sum(H3K27me3))

write.csv(newdata, ".../dadosTratados.csv", row.names = FALSE)

train<-read.csv(file=".../dadosTratados.csv",header=T,sep=",")
```

Métodos – Calcular Covariância e Correlação

Antes de aplicar o PCA, temos que garantir que as características são dependentes e altamente correlacionadas. Para isso, calculamos a covariância e o coeficiente de correlação, respectivamente.

Estes cálculos foram realizados usando as funções *cov () e cor ()* em R.

- **Resultado da Covariância**: características dependentes (valores diferentes de zero)

	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

Métodos – Calcular Covariância e Correlação

- **Resultado da Coeficiente de Correlação**: foi calculado o coeficiente de correlação de Pearson, variando de -1 a 1. Quanto mais próximo de 1, maior a correlação.

	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

Métodos – PCA

Como as características possuem dependência e uma boa correlação, podemos aplicar o PCA.

Temos dois métodos de PCA para aplicar: *Spectral Decomposition e Singular Value Decomposition.* O primeiro analisa a covariância e correlação das características, e o segundo dos dados. O segundo é indicado para conjuntos de dados onde o número de características é maior do que instâncias e tem mais precisão para calcular a acurácia.

No caso do conjunto de dados usado, será aplicado o PCA usando o *Spectral Decomposition*. Em R há dois pacotes para este método: *eigen* e *princomp*. Usaremos o *princomp*.

Métodos – PCA

O *princomp* pode ser aplicado usando a matriz de correlação ou a matriz de covariância.

- Usando matriz de correlação

O parâmetro cor define o uso da matriz de correlação, e o scores são os valores dos componentes para cada instância do conjunto de dados.

```
#PCA usando princomp com matriz de correlação
pca.cor<-princomp(train[,-c(1)], cor=TRUE, scores = TRUE)
biplot (pca.cor)
screeplot(pca.cor)
View(pca.cor)
write.csv(pca.cor$scores, ".../pca_correlacao.csv", row.names = FALSE)
```

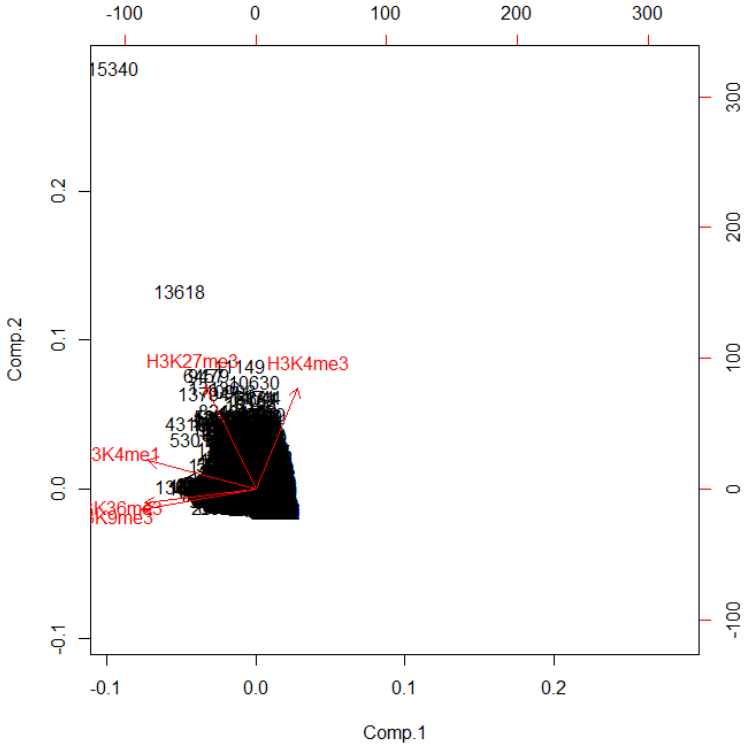
Métodos – PCA

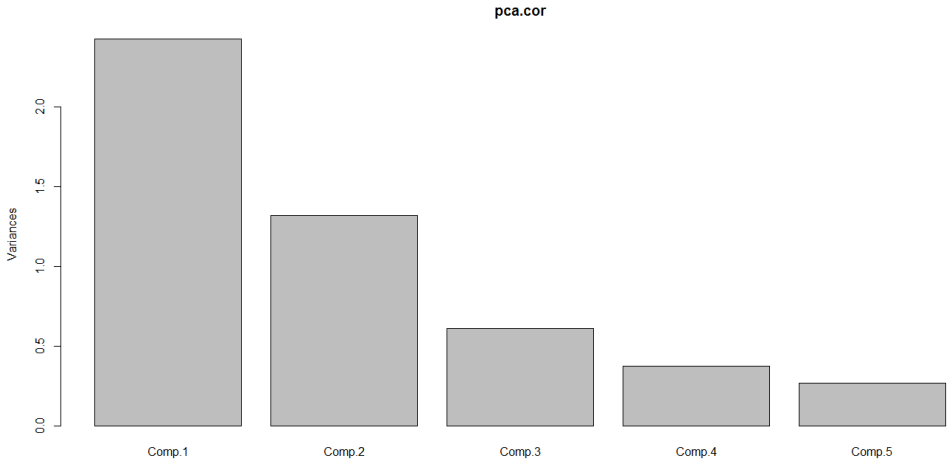
Informações obtidas após execução do script:

pca.cor	list [7] (S3: princomp)	List of length 7
sdev	double [5]	1.557 1.148 0.783 0.612 0.518
Comp.1	double [1]	1.557484
Comp.2	double [1]	1.148437
Comp.3	double [1]	0.7828178
Comp.4	double [1]	0.6117197
Comp.5	double [1]	0.5180055
loadings	double [5 x 5] (S3: loadings)	0.209 -0.531 -0.546 -0.562 -0.246 0.677 ...
center	double [5]	163 153 337 510 112
H3K4me3	double [1]	162.8247
H3K4me1	double [1]	153.3522
H3K36me3	double [1]	336.8313
H3K9me3	double [1]	509.966
H3K27me3	double [1]	111.8989
scale	double [5]	135.6 85.7 249.5 480.7 61.9
H3K4me3	double [1]	135.6455
H3K4me1	double [1]	85.66053
H3K36me3	double [1]	249.4813
H3K9me3	double [1]	480.7353
H3K27me3	double [1]	61.92722
n.obs	integer [1]	15485
scores	double [15485 x 5]	1.4847 1.9363 -2.7707 -1.5021 -0.9723 0.9666 -0.3264 -0.5750 -0.1200 0.2902 ...

Métodos – PCA

Após aplicar o PCA:





Métodos – PCA

Como temos mais de 3 dimensões, não é possível visualizar com precisão a distribuição dos dados. Isto implica em usar a classificação com e sem PCA e comparar os resultados.

Para escolher os componentes, usamos o critério de maior variabilidade, tendo 0.5 como limiar. Desta forma, foram escolhidos 3 componentes:

- *Comp1, Comp2, Comp3.*

Após a escolha, foi gerado um novo conjunto de dados:

Antes: 15485 x 5 – instâncias x características originais (dadosTratados.csv)

Depois: 15485 x 3 – instâncias x componentes (pcaCorTratado.csv)

Métodos – PCA

- Usando matriz de covariância

O parâmetro cor define o uso da matriz de correlação, e o scores são os valores dos componentes para cada instância do conjunto de dados.

```
#PCA usando princomp com matriz de covariancia
pca.cov<-princomp(train[,-c(1)], cor=FALSE, scores = TRUE)
biplot (pca.cov)
screeplot(pca.cov)
View(pca.cov)
write.csv(pca.cov$scores, ".../pca_covariancia.csv", row.names = FALSE)
```

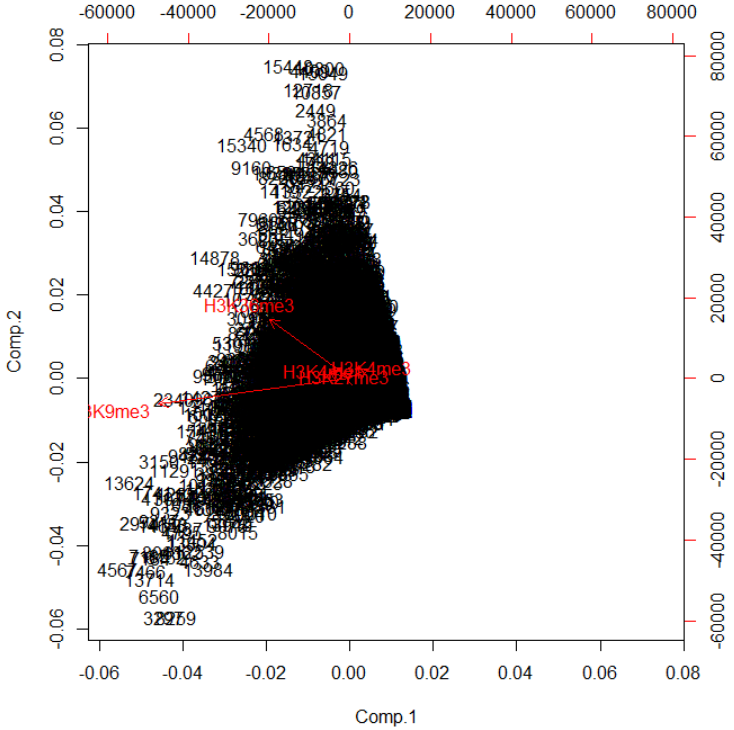
Métodos – PCA

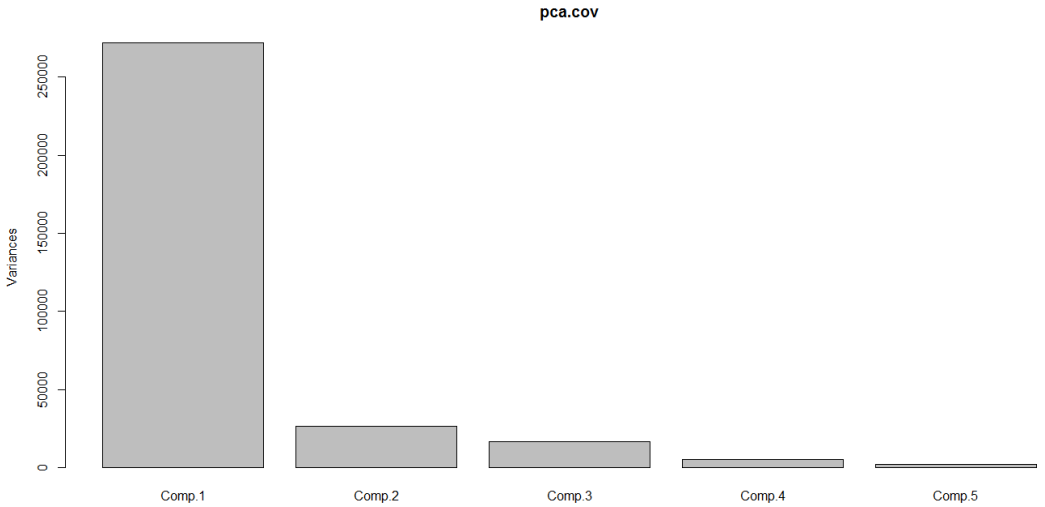
Informações obtidas após execução do script:

pca.cov	list [7] (S3: princomp)	List of length 7
sdev	double [5]	521.2 163.0 130.3 74.4 46.5
Comp.1	double [1]	521.2389
Comp.2	double [1]	162.9949
Comp.3	double [1]	130.2631
Comp.4	double [1]	74.35655
Comp.5	double [1]	46.53061
loadings	double [5 x 5] (S3: loadings)	0.0838 -0.0988 -0.3838 -0.9140 -0.0214 0.1430 ...
center	double [5]	163 153 337 510 112
H3K4me3	double [1]	162.8247
H3K4me1	double [1]	153.3522
H3K36me3	double [1]	336.8313
H3K9me3	double [1]	509.966
H3K27me3	double [1]	111.8989
scale	double [5]	1 1 1 1 1
H3K4me3	double [1]	1
H3K4me1	double [1]	1
H3K36me3	double [1]	1
H3K9me3	double [1]	1
H3K27me3	double [1]	1
n.obs	integer [1]	15485
scores	double [15485 x 5]	395.904 543.540 -859.368 -583.598 -8.307 420.646 -82.091 -115.829 92.247 ...

Métodos – PCA

Após aplicar o PCA:





Métodos – PCA

Usando a matriz de correlação, a melhor redução de dimensionalidade seria de 5 para 2 ou 3 componentes.

Usando a matriz de covariância, a melhor redução de dimensionalidade seria de 5 para 1 componentes apenas, devido a alta variabilidade comparado aos outros componentes.

- Diante disso, foi escolhido apenas o *Comp1.*

Após a escolha, foi gerado um novo conjunto de dados:

Antes: 15485 x 5 – instâncias x características originais (dadosTratados.csv)

Depois: 15485 x 1 – instâncias x componentes (pcaCovTratado.csv)

Discussão

Aplicando PCA usando a matriz de correlação, reduzimos em 40% a dimensionalidade, e usando a matriz de covariância, reduzimos em 80% a dimensionalidade.

Como os valores das características de cada instâncias no conjunto original estão normalizados, podemos descartar o resultado do PCA com a matriz de correlação, e aplicar na classificação o resultado do PCA com a matriz de covariância.

Em ambos os casos, o recomendado é realizar a classificação usando as características vs os dois resultados do PCA, pois temos mais de uma dimensão no conjunto de dados, e validar os resultados.

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

3. Seleção de Características

SIN5007- Reconhecimento de Padrões

Profa. Ariane Machado Lima

Setembro, 2017

Introdução Sobre Seleção de Características

Seleção de características também é uma forma de fazer redução da dimensionalidade.

Para fazer seleção de características, vamos usar o método heurístico **Weighting.** Este método percorre o espaço de caraterísticas **com base em pesos**, onde são passados pesos às características para comparar a um limiar e assim, selecionar um subconjunto de características.

Os pesos auxiliam na escolha do melhor subconjunto, por entrar com uma métrica de seleção, além de considerar todas as características. O que não ocorre nos métodos heurísticos **incrementais**, onde as características relevantes podem não ser consideradas no processamento, não garantindo o subconjunto ótimo de características.

Introdução Sobre Seleção de Características

Um exemplo de algoritmo heurístico Weighting é o Relief.

- Temos a versão **Relief** que foi desenvolvido para seleção de características para classificação binária e usando norma euclidiana, encontrando o NearHit e o NearMiss e recomendado rodar várias vezes;
- O **ReliefF** é uma versão melhorada do Relief, usando norma Manhattan e buscando valores absolutos de NearMiss e NearHit e recomendado rodar de forma exaustiva para uma quantidade pequena de instâncias (~1000) e pode ser usado para classificação multi-classes;
- O **RReliefF** é uma versão melhorada do ReliefF. Foi adaptado para ser usado em problemas de regressão.

Objetivo

Aplicar o algoritmo **Relief** ao conjunto de dados, a fim de obter o melhor subconjunto de características para classificação das instâncias, e poder comparar os resultados com o PCA.

Métodos

- **Rotina utilizada**

Usando R, temos dois pacotes do Relief: Dprep e FSelector.

O **Dprep** (*Data Preprocessing for supervised classification*) possui a função Relief(), que segue a lógica da primeira versão.

O **FSelector** (*Package for selecting attributes*) possui a função Relief(), que segue a lógica da última versão - ***RReliefF***.

Vamos aplicar o pacote Fselector, por considerar a última versão do Relief, e porque o algoritmo busca os pesos em atributos contínuos e discretos.

Métodos

Sintaxe da função Relief:

relief(formula, data, neighbours.count = X, sample.size = X)

formula: coluna com a classificação

data: conjunto de dados

neighbours.count: número de vizinho à encontrar para cada instância

sample.size: número de instâncias

Métodos

O método heurístico considera a classificação das instâncias. Por isso, antes de aplicar o Relief ao conjunto de dados, foi necessário um pré-processamento.

Foi feito um merge dos arquivos **dadosTratados.csv** *vs* **y_train**, a fim de obter um novo arquivo com todas as características, instâncias, id e classe.

```
classes<-read.csv(".../y_train.csv")
dados_relief <- merge(train,classes)
View(dados_relief)
write.csv(dados_relief, ".../dadosRelief.csv", row.names = FALSE)
dados_relief <- dados_relief[,-c(1)]
```

Métodos

Finalizado a etapa de pré processamento, rodamos a função do Relief.

Consideramos os resultados com base na busca por 10 vizinhos para 20 instâncias e 20 vizinhos para 60 instâncias, por questão de tempo de execução do algoritmo.

Executando para encontrar 120 vizinhos para todo o conjunto de dados (15.485 instâncias), ultrapassou duas horas de execução.

```
pesos<-relief(Prediction~., dados_relief, neighbours.count = 10, sample.size = 20)
print(pesos)
subset <- cutoff.k(pesos, 3)
f <- as.simple.formula(subset, "Prediction")
print(f)
```

Métodos

Os pesos encontrados para cada característica (a cada execução, mudam os valores dos pesos, usando a mesma quantidade de vizinhos e instâncias):

	attr_importance
H3K4me3	1.988957e-03
H3K4me1	1.802091e-03
H3K36me3	-2.203641e-04
H3K9me3	-7.740269e-04
H3K27me3	-3.314459e-05

	attr_importance
H3K4me3	-0.0008628798
H3K36me3	-0.0011558492
H3K27me3	-0.0014368242
H3K9me3	-0.0019093711
H3K4me1	-0.0022241585

	attr_importance
H3K4me1	-2.129713e-03
H3K9me3	-1.949916e-03
H3K36me3	-1.579099e-03
H3K4me3	-3.370581e-04
H3K27me3	-5.978576e-05

Mesmo havendo variação de pesos a cada execução, os resultados não mudaram quanto ao ranking de melhor subconjunto de características.

O tempo de execução varia de acordo com o número de instâncias para explorar, vizinhos a encontrar e características. Porém é mais lento do que o Princomp (PCA)

Resultados

Foi executado o algoritmo buscando as 3 melhores características e gerado um novo conjunto de dados. Como no PCA escolhemos 3 componentes, para manter uma proporção nas validações dos métodos, aplicamos 3 também ao Relief.

result	formula	Prediction ~ H3K4me3 + H3K36me3 + H3K27me3
[[1]]	symbol	`~`
[[2]]	symbol	`Prediction`
[[3]]	language	H3K4me3 + H3K36me3 + H3K27me3
[[1]]	symbol	`+`
[[2]]	language	H3K4me3 + H3K36me3
[[3]]	symbol	`H3K27me3`

Resultados

Características iniciais:

- H3K4me1
- H3K36me3
- H3K27me3
- H3K9me3
- H3K4me3

Após o Relief:

- H3K36me3
- H3K27me3
- H3K4me3

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

4. K Fold Cross Validation

SIN5007- Reconhecimento de Padrões

Profa. Ariane Machado Lima

Outubro, 2017

Introdução

Uma forma de avaliar o classificador é calculando a taxa de erro.

No caso do aprendizado supervisionado, podemos aplicar alguns métodos ao conjunto de dados para estimar o erro.

Exemplos: Resubstituição, Holdout, Leave-one-out, K-fold cross validation, Bootstrap.

Dentre estes exemplos, uma melhor opção é o K-fold cross validation, por ser menos enviesada que o holdout, menos custoso que o leave-one-out. O método de resubstituição é uma má opção, pois usa o mesmo conjunto para treinamento e teste, e o Bootstrap é indicado para pequenos conjuntos de dados e é computacionalmente caro.

K Fold Cross Validation

O cálculo do erro usando este método consiste:

ErroTotal = 1/K **Σ** n até k ErroFold-n

O erro é o produto da divisão de 1 por K (partes) pelo somatório do erro em cada execução.

Por exemplo, se o data set foi dividido em 5 partes (K), executamos 5x onde em cada execução usamos K-1 para treino e a última parte para teste (k – [k-1]). Calculamos o erro e aplicamos no cálculo do erro total, que é a estimativa de erro do classificador.

Objetivo

O objetivo é implementar em R uma função que irá fazer a divisão do conjunto de dados original, conjunto de dados com PCA e conjunto de dados com seleção de características em K partes, para que em cada execução, usarmos K-1 para treinar e a parte que ficou de fora para testes.
Pré- requisitos:

1. Cada parte deve ter a diferença de, no máximo 1 elemento;
2. Cada parte deve manter a proporção de cada classe;
3. Imprimir quantos elementos ficou em cada parte: de cada classe e total.

Métodos

- Foi feito um algoritmo para fazer a divisão dos folds. O script está no arquivo Script_Final.r.

Resultados

Total de subconjuntos: 5

	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

Discussão

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:** 1/K **Σ** n até k ErroFold-n (significa o quanto que o classificador errou)

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

5. SVM

SIN5007- Reconhecimento de Padrões

Profa. Ariane Machado Lima

Novembro, 2017

Máquina de vetores de suporte - SVM

SVM é um classificador para aprendizado supervisionado, onde tem associado uma função de decisão: $D(x) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$

Na fase de treinamento, usamos SVM para definir os limiares para classificação.

Na fase de classificação ou testes, usamos SVM para definir em qual “lado” do limiar a instância está e assim fazer a classificação.

Objetivo

Aplicar o SVM para classificar o conjunto de dados, implementando em R, usando o pacote e1071.
Usar os dados originais, com PCA e com Relief para comparar resultados.

Métodos

- **Rotina utilizada**

Foi utilizada a função svm(), do pacote {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.

Obs.: foi aplicado os dados com PCA usando prcomp.

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

6. Redes Neurais Artificiais

SIN5007- Reconhecimento de Padrões

Profa. Ariane Machado Lima

Novembro, 2017

Introdução

Redes neurais artificiais são modelos computacionais baseados no sistema nervoso central. Podem ser usadas no aprendizado supervisionado para fazer classificação de dados. O algoritmo de uma rede neural para convergir, é preciso ir fazendo ajustes dos pesos (sinapses).

Há vários modelos de arquitetura de redes neurais. Uma delas é o Perceptron, um dos mais antigos, do tipo FeedForward, podendo ter uma camada escondida ou multicamadas(MLP).

RNA - Perceptron

Para fazer classificação dos dados usando redes neurais, vamos usar o Perceptron com uma camada escondida, por ser um modelo de rede neural simples para treinar.
Este modelo por ser FeedForward, não tem realimentação da rede e nem propagação do erro. Basicamente, o algoritmo tem entrada dos dados (camada 01), processamento (camada escondida) e saída (camada 02).
A implementação será feita em R, usando a função *nnet()*.
O conjunto de dados para usar no treinamento devem estar com os rótulos.

Objetivo

Aplicar o algoritmo Perceptron para classificar o conjunto de dados:

- Usando o conjunto de dados original
- Usando o conjunto de dados resultado do PCA – matriz de correlação
- Usando o conjunto de dados resultado do PCA – matriz de covariância
- Usando o conjunto de dados resultado da Seleção de Características
- Comparar os resultados com o SVM e NaiveBayes (tempo de execução, taxa de erro etc)

Métodos

- **Rotina utilizada**

Foi utilizado o pacote nnet () do R.

As execuções com redes neurais foram realizadas variando: máximo de interações (100 e 500), número de neurônios (2, 5 e 10), pesos iniciais (de 0.1 a 0.8), taxa de aprendizado (0.1 a 2), valor máximo para os pesos.

```
library(nnet)
install.packages("confusionMatrix")
library(caret)
fold <- read.csv(file="../fold_X.csv",header=T,sep=",")
View(fold)
result_nnet1 <- nnet(Prediction~., fold, rang=0.1,size=2, maxit=500, subset=NULL, Hess=TRUE, na.action=NULL,
MaxNWts=1000,contrasts=NULL, linout = FALSE, decay = seq(from = 0.1, to = 0.5, by = 0.1))
View(result_nnet1)
#result_nnet1 <- nnet(Prediction~.,fold,weights=NULL, size=2, Wts=100,rang = 1, decay = 0, maxit = 100,
Hess = TRUE, trace = TRUE, MaxNWts = 1000, abstol = 1.0e-4, reltol = 1.0e-8)
results1 <- predict(result_nnet1, newdata=fold)
conf1 <- confusionMatrix(results1, fold$Prediction)
table(factor(pred, levels=min(test):max(test)), factor(test, levels=min(test):max(test)))
table(fold$Prediction, predict(result_nnet1, fold, type = "class"))
```

Discussão

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.

Resultados

```
> 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
```

Resultados

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

Resultados

Quanto a matriz de confusão, também encontramos dificuldades técnicas para fazer a geração usando o R.

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

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

7. Naive Bayes

SIN5007- Reconhecimento de Padrões

Profa. Ariane Machado Lima

Novembro, 2017

Introdução

- Script_Final.pdf
- Tabela de Resultados.pdf
- Atividade Final.pdf (Slides usados na apresentação)

Objetivos, Métodos e Resultados

Da mesma forma que aplicamos SVM e Redes Neurais, replicamos usando o 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)

Objetivos, Métodos e Resultados

O Naive Bayes executou rápido, principalmente comparado ao SVM.

Porém, em termos gerais de classificação, teve o pior desempenho.

Mais detalhes na “conclusão” e no arquivo “Tabela de Resultados”.

Apêndice

- Script_Final.pdf
- Tabela de Resultados.pdf
- Atividade Final.pdf (Slides usados na apresentação)

Conclusão

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.

Conclusão

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.

Conclusão

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	0.000129449838

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

Trabalhos Futuros

- Melhorar (e muito) a execução de Redes Neurais para o conjunto de dados;
- Aplicar os classificadores novamente usando os dados do PCA com o princomp.

Referências

https://inclass.kaggle.com/c/gene-expression-prediction

https://dadosdadosdados.wordpress.com/2015/06/24/tutorial-r-organizando-dados-com-dplyr-e-tidyr-2/

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/cor.html

https://www.ime.usp.br/~pavan/pdf/MAE0330-PCA-R-2013

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

Referências

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

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

https://www.rdocumentation.org/packages/nnet/versions/7.3-12/topics/nnet