AGA 0505 - Análise de Dados em Astronomia

10. Aprendizado de Máquina: Regressão e Classificação

Laerte Sodré Jr.

1o. semestre, 2023

aula de hoje:

1. o que é regressão
2. regressão linear ordinária
3. modelos e generalização
4. regularização
5. regressão com kernel
6. regressão com k-nn
7. o que é classificação
8. a função custo em classificação
9. completeza e contaminação
10. regressão logística
11. árvores de decisão
12. random forest
13. support vector machine

Modelos devem ser o mais simples possível, não mais que isso

(atribuido a Einstein)

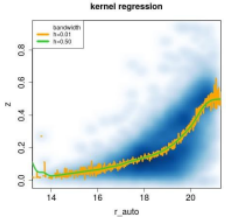
o que é regressão

regressão em ML

- estimativa de uma variável contínua, y, a partir de dados x em um conjunto de treinamento
- regressão é um tipo de *aprendizado supervisionado*
- exemplos:
  - redshifts fotométricos: $y_{train} = z_{spec}$, $x_{train} = (u, g, r...)$
  - massa estelar de uma galáxia: $y_{train} = M_\star$, $x_{train} = (z_{phot}, u, g, r...)$
- regressão envolve *estimativa de parâmetros*:

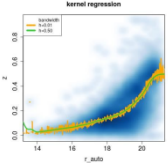
$$y = f(x; w)$$

- muitos modelos para $f(x; w)$: regressão linear, kernel, k-nn, ANN, ...
- $f(x; w)$ pode ser considerada uma *caixa preta*
- o desempenho de diferentes modelos pode ser avaliado com um *conjunto de teste*



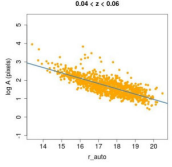
o que é regressão

- modelo: $y = f(x; w) + \epsilon$
  w: parâmetros do modelo
  ϵ: erro ou ruído
- há muitas possíveis fontes de erro:
  - ϵ pode ser devido a erros de medida em x e/ou y
  - ϵ pode ser devido a inadequações do modelo (muito simples, muito complicado) - *erros epistêmicos*



o que precisamos para treinar um modelo:

- o modelo $y = f(x; w)$
- $l(w)$- função de custo ou de perda:
  ela é uma métrica da qualidade do ajuste do modelo
- otimizador: algoritmo que busca w pela *minimização* de $l(w)$
  ex.: descida do gradiente

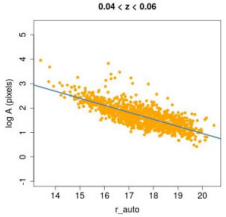


exemplo: regressão linear ordinária

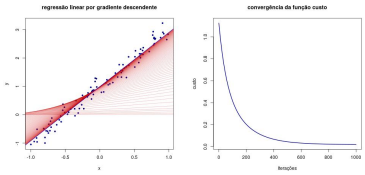
- *modelo linear*: $y = w_0 + w_1 x + \epsilon$
  linear nos parâmetros: $\{w_0, w_1\}$
- função de custo:
  soma dos quadrados dos resíduos:

$$l(w) \propto \chi^2 = \sum_{i=1}^{N} \frac{[y_i - (w_0 + w_1 x_i)]^2}{\epsilon^2}$$

- otimização: minimização do χ^2

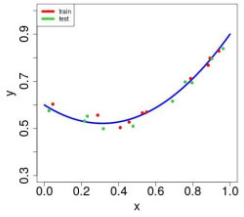


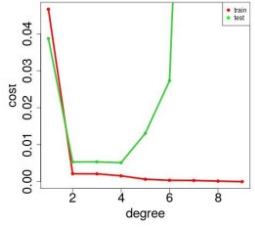
- aprendizagem por descida do gradiente
  - atualização iterativa de w:
    $w \longleftarrow w - \lambda \times \nabla l(w)$
  - para um certo dado i:
    - $w \longleftarrow w + 2\lambda[y_i - y(x_i; w)]/\epsilon^2$
      o termo $[...]$ é o resíduo de i com o valor corrente de w
    - com o treinamento, os resíduos diminuem e convergem para um valor estacionário



regularização

- modelos devem ter a “complexidade” (ou “capacidade”) correta
- modelos muito simples: *underfitting*
- modelos muito complexos: *overfitting* $\longrightarrow$ os modelos ajustam o ruído!
- ex- ajuste de um polinômio de grau M: o modelo é ajustado a um conjunto de treinamento e aplicado a um conjunto de teste
- o que acontece com valores grandes de M? overfitting! o modelo ajusta o ruído: w “explode”



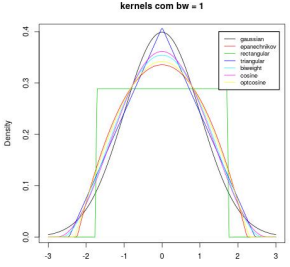


regularização

- com valores grandes de M, w "explode"
- ex: para $M = 9$ (página anterior):
  30.87, -1122.61, 13019.71, -75589.66, 256956.84, -544754.35, 730907.60, -603984.86, 280679.44, -56144.84
- uma forma de prevenir *overfitting*:
  regularização- restringe o valor dos parâmetros
- ex: adição de um termo na função de custo:
  $l(w) = \chi^2/2 + \alpha w^T.w$
  α: (hiper)parâmetro de regularização
- o termo adicional penaliza grandes valores absolutos de w
- modelo linear: *ridge regression*
  $\mathbf{W} = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$
- LASSO: *least absolute shrinkage and selection*
  $l(w) = \chi^2/2 + \alpha|w|$
- note que a regressão linear ordinária é um caso particular de LASSO e da *ridge regression*
- α pode ser determinado por *validação cruzada*
- regularização é útil quando se tem muitas variáveis ou se essas variáveis são correlacionadas

regressão com kernel

- kernel: tipo de função que calcula médias *locais* no espaço de dados
- $K(x)$: kernel
  hiperparâmetro h: *bandwidth*
- existem muitos tipos de kernel:

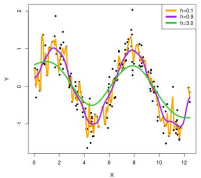


- regressão com kernel: tipo de regressão *local*:

$$y = f(x) = \frac{\sum_{i=1}^{N} K(|x - x_i|/h) y_i}{\sum_{i=1}^{N} K(|x - x_i|/h)} = \sum_{i=1}^{N} w_i(x) y_i$$

- y: média ponderada "local" dos valores de y com pesos $w_i(x)$

$$w_i(x) = \frac{K(|x - x_i|/h)}{\sum_{j=1}^{N} K(|x - x_j|/h)}$$



regressão com k-NN

- $k-NN$: algoritmo *k-th nearest neighbor*
  - $y(x)$ é a média ou a média ponderada dos valores de y dos k pontos mais próximos de x
  - útil para regressão e classificação
  - hiperparâmetro k
  - implementa a intuição de que valores próximos de x devem ter valores próximos de y

- distância entre dois pontos:
  - ponto em d dimensões: $\mathbf{x}_i = \{x_{i1}, x_{i2}...x_{id}\}$
  - métrica - ex.: distância de Minkowski

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^{d} (|\mathbf{x}_i - \mathbf{x}_j|^p \right)^{1/p}$$

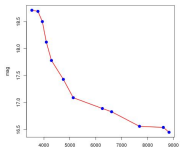
$p = 1$: distância Manhattan;
$p = 2$: distância Euclidiana

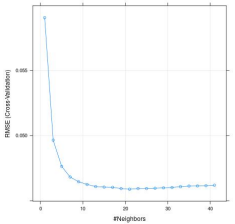
- a escala dos dados é importante!
- a distância a ser usada depende do tipo e dimensão dos dados

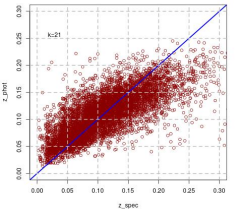
regressão com k-NN

- exemplo: estimativa de redshift fotométrico
  conjunto de treinamento:
  - input: magnitudes nas 12 bandas do S-PLUS
  - output: redshift espectroscópico do SDSS
- $k - NN$ não é muito preciso, mas é rápido e provê uma boa referência para comparação com outros modelos

exemplo de um *foto-espectro* do S-PLUS (input)



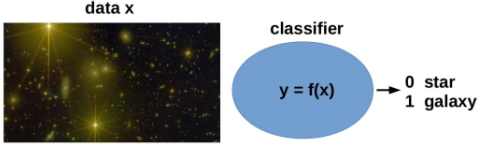




o que é classificação

- classificação: $y = f(x; w)$
- x pode ser uma variável real ou categórica
- y: variável *qualitativa/categórica/discreta*
- y representa classes ou alvos:
  a classificação pode ser binária ou multiclasse
- exemplos:
  - classificação morfológica de galáxies em 4 classes: E, S0, S, Ir
  - classificação estelar em 7 classes: OBAFGKM
  - 2 classes: estrela/galáxia
- *em classificação os conjuntos de treinamento devem ser balanceados!*

- *one hot encoding*:
  - no conjunto de treinamento, os alvos y são vetores com dimensão igual ao número de classes
  - os alvos y são iguais a 0, exceto o que corresponde à classe correta, que é 1
    ex.: (0,1) or (1,0) para classificação binária
    (1,0,0), (0,1,0), (0,0,1) para 3 classes ...



a função custo para classificação binária

- custo “0-1”:
  - atribui-se 1 a uma classificação errada e 0 a uma certa
  - se $\hat{y}$ é a estimativa de y, o custo é

$$L(y, \hat{y}) = \begin{cases} 1 & \text{se } \hat{y} \neq y \\ 0 & \text{se } \hat{y} = y \end{cases}$$

  - *risco* da classificação = taxa de erro

$$E[L(y, \hat{y})] = P(\hat{y} \neq y)$$

- entropia cruzada

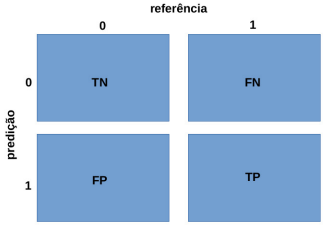
$$CE(y, \hat{y}) = -\frac{1}{N} \sum_i [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

- erro quadrático medio:

$$\frac{1}{N} \sum_i (\hat{y}_i - y_i)^2$$

- classificação binária (0,1)
  - TN: true negative
  - TP: true positive
  - FN: false negative
  - FP: false positive

matriz de confusão



completeza e contaminação (2 classes)

- completeza: fração de detecções (*recall)*

$$R = \frac{TP}{TP + FN}$$

- contaminação: fração de detecções erradas

$$\frac{FP}{TP + FP}$$

- acurácia: fração de detecções corretas

$$\frac{TP + TN}{TP + TN + FP + FN}$$

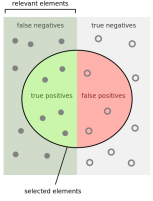
- eficiência ou precisão = 1 - contaminação

$$P = \frac{TP}{TP + FP}$$

- F_1 score:

$$F_1 = 2 \times \frac{P \times R}{P + R}$$

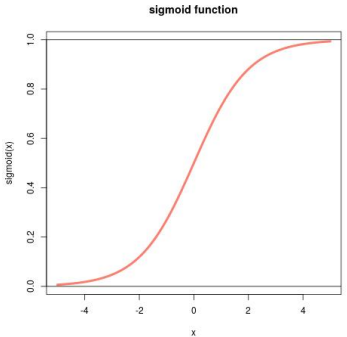
- dependendo da natureza do problema e do objetivo, pode-se otimizar a completeza ou a eficiência ou F_1, etc



exemplo: regressão logística

- regressão logística: apesar do nome é usada para classificação
- função sigmoide:

$$f(x) = \frac{1}{1+e^{-x}}, \qquad 0 < f < 1$$



- suponha que temos objetos num conjunto de treinamento com atributos x (ex..,FWHM, magnitudes, ...) classificados como estrelas, $y =$"0", ou galáxias, $y =$"1"
- a regressão logística modela a probabilidade de um objeto ser uma galáxia como:

$$P(y = 1|\mathbf{x}, \mathbf{w}) = p = \frac{1}{1+e^{-\mathbf{w}.\mathbf{x}}}$$

- usamos o conjunto de treinamento para estimar os parâmetros $\mathbf{w}$ e então predizemos as classes dos objetos no conjunto de teste
($\mathbf{x}$ e $\mathbf{w}$ podem ser vetores)

função de custo da regressão logística

$$P(y=1|\mathbf{x},\mathbf{w}) = p(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}.\mathbf{x}}}$$

- verossimilhança dos parâmetros do modelo:
  - probabilidade de '1': p
  - probabilidade de '0': $1-p$
  - y_i: 0 ou 1
  - probabilidade p_i:

$$p_i^{y_i}(1-p_i)^{1-y_i}$$

  - verossimilhança:

$$\mathcal{L}(\mathbf{w}) = \Pi_i \, p_i^{y_i}(1-p_i)^{1-y_i}$$

- função de custo:

$$-\log \mathcal{L} = -\sum_i \left[y_i \log p_i + (1-y_i)\log(1-p_i) \right]$$

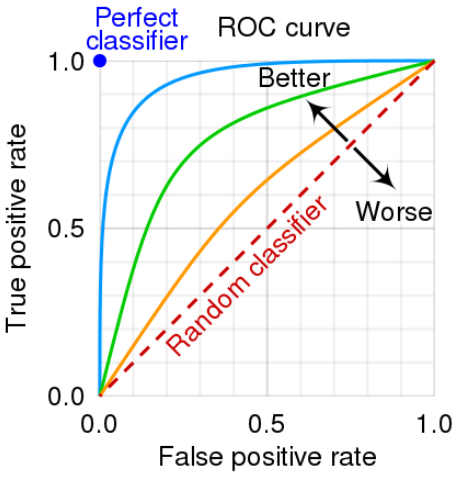
(entropia cruzada binária)

- os parâmetros podem ser estimados por máxima verossimilhança: $\hat{\mathbf{w}}$
- predição para um novo x:
  se $p(x) > 0.5$: o objeto é classificado como galáxia
  se $p(x) < 0.5$: como uma estrela

- a regressão logística é útil em problemas linearmente separáveis

classificação binária

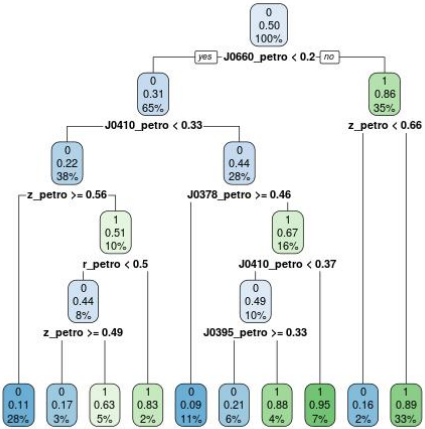
- curva ROC:
  - ROC: *receiver operating characteristic*
  - FP x TP
  - útil para tomada de decisão
- estatística útil:
  AUC: *area under the curve*



árvores de decisão (*decision trees*)

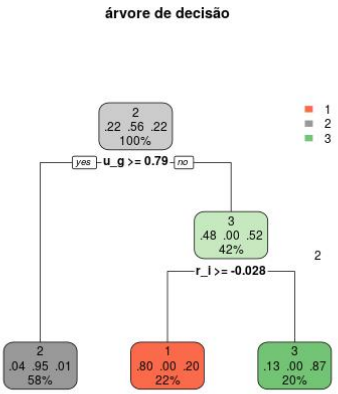
- AD: úteis para regressão e classificação
- AD segmenta o espaço de dados do input em várias regiões simples
- a predição y para um novo dado depende da região a que ele pertence:
  - regressão: média dos valores de y
  - classificação: classe mais provável
- vantagem: interpretável
- desvantagem: pouco competitiva!
  mas métodos de *ensemble*, que combinam muitas ADs, são poderosos: *bagging, random forest, boosting*

classificação estrela/galáxia a partir da fotometria do S-PLUS:



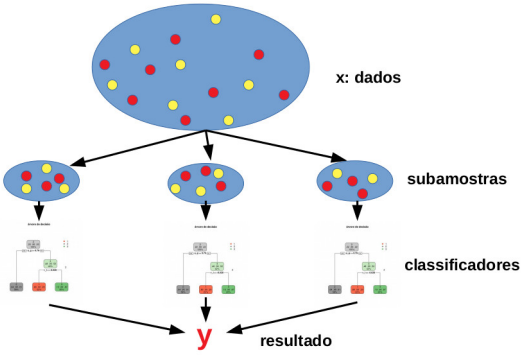
árvores de decisão

- AD: regras de particionamento dos dados, começando no topo da árvore
- AD: nós, ramos, folhas (nós terminais)
- em cada nível da árvore, os nós são divididos em dois (ou mais) ramos, de acordo com um limite de decisão
- a AD cresce (de cima para baixo) até que um certo critério seja satisfeito
- as folhas registram os pontos de cada nó com seus valores de y
- novo dado: segue-se os nós da árvore em uma série de decisões binárias até uma folha



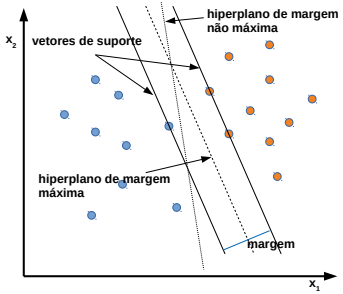
random forest/floresta aleatória

- tipo de *ensemble learning*: combinação dos resultados de muitos modelos
- RF combina muitas ADs, cada uma usando apenas uma fração aleatória dos atributos de entrada
- se os dados têm M atributos, cada AD usa apenas $m < M$ atributos
- parâmetros: número de árvores (n) e atributos (m) de cada árvore
- m relativamente pequeno evita *overfitting* e melhora as predições
- n e m podem ser escolhidos por validação cruzada



SVM: *support vector machines*

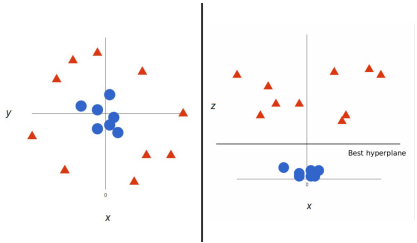
- um dos melhores algoritmos para classificação
- *classificador de margem máxima*: classificação binária usando o hiperplano de margem máxima
  - margem: distância mínima dos dados ao hiperplano
  - hiperplano de margem máxima: o hiperplano que separa as classes para o qual a margem é máxima
  - vetores de suporte: dados que "suportam" o hiperplano de margem máxima
- o classificador de margem máxima é útil para problemas linearmente separáveis



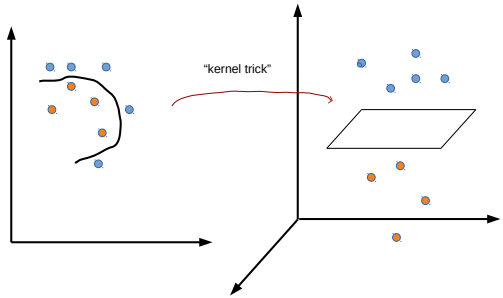
- e quando os dados não são linearmente separáveis?

SVM: *support vector machines*

- pode-se classificar dados não separáveis linearmente aumentado o espaço de atributos
- exemplo: na figura abaixo, a introdução de uma nova variável torna as classes separáveis
  $z = x^2 + y^2$



- SVM: aumenta o espaço de atributos usando kernels e margens *soft*:
  permite-se alguma mistura de classes a custa de uma penalização



dicas para projetos de ML

(baseado parcialmente no *the universal workflow of ML*, de Chollet)

1. defina a tarefa:
   - qual é o problema a ser resolvido?
   - que dados serão usados?
     esta é a parte mais difícil (!) e demorada da maioria dos projetos de ML;
     os dados devem ser representativos da aplicação desejada
   - que tipo de algoritmo pode ser usado?
   - que métrica(s) pode ser adotada para avaliar o resultado?
2. desenvolva o modelo:
   - pré-processamento dos dados: normalização, vetorização, dados faltantes
   - adote um protocolo para avaliar as métricas de sua aplicação: selecione um procedimento para treinamento usando conjuntos de treinamento representativos
   - comece simples: desenvolva um modelo simples (e rápido) para avaliar e selecionar a) as variáveis de entrada, b) a configuração do algoritmo, c) a estratégia de treinamento
   - seja ousado: desenvolva um modelo que overfita: *o modelo ideal está no limite entre underfitting e overfitting, e, para saber onde este limite está, você tem que ultrapassá-lo*
   - regularize e ajuste o modelo para maximizar a generalização

     (isso também toma tempo!)
3. aplique o modelo:
   - analize os pontos fracos do modelo: ex., viéses, classificações erradas, ...
   - apresente o desempenho final