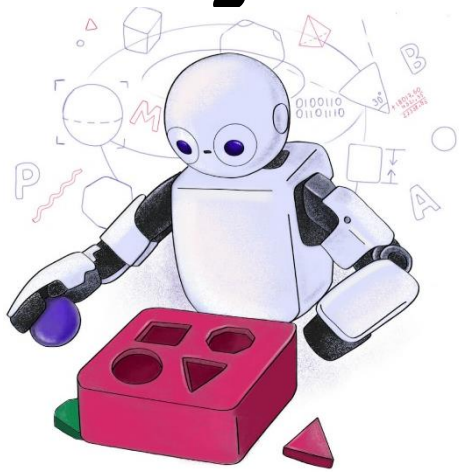
TP555 - Inteligência Artificial e Machine Learning:

Classificação Linear



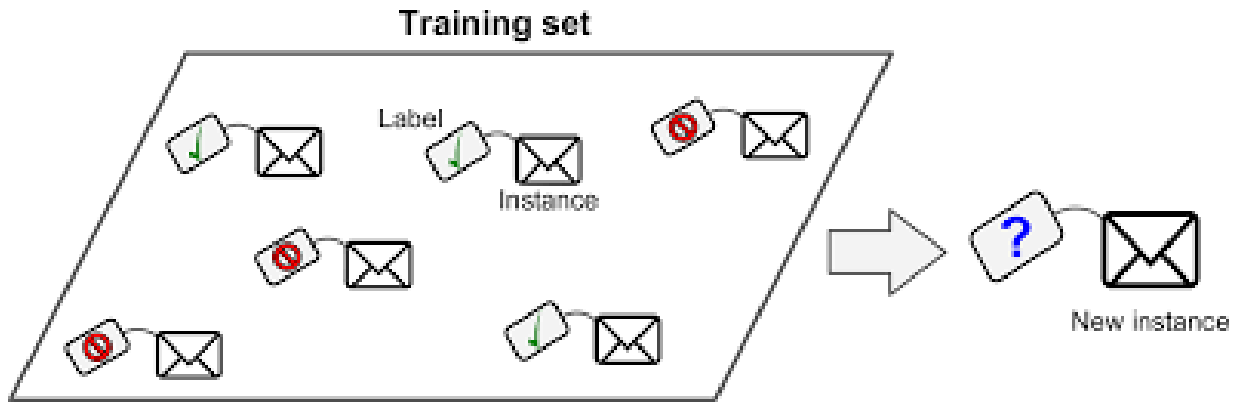
Inatel

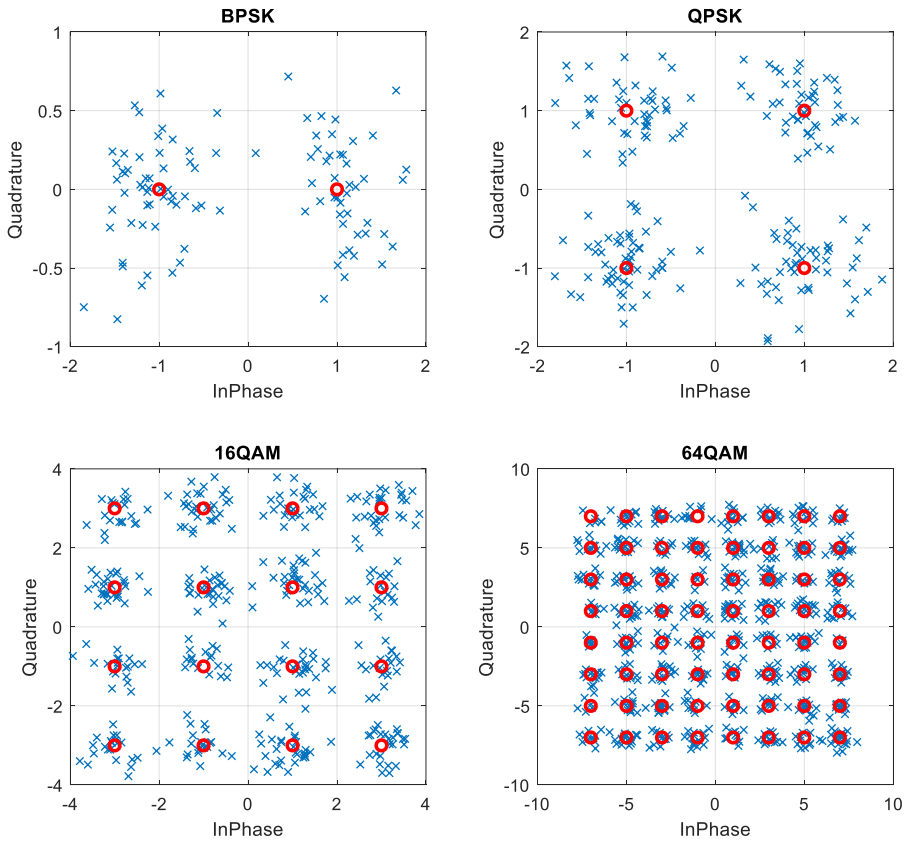
Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

Tópicos abordados

- Abordagens para classificação linear:
  - Classificação Bayesiana (Aula de hoje)
  - Regressão logística (Próxima aula)
- Métricas para avaliação de classificadores (Próxima aula).

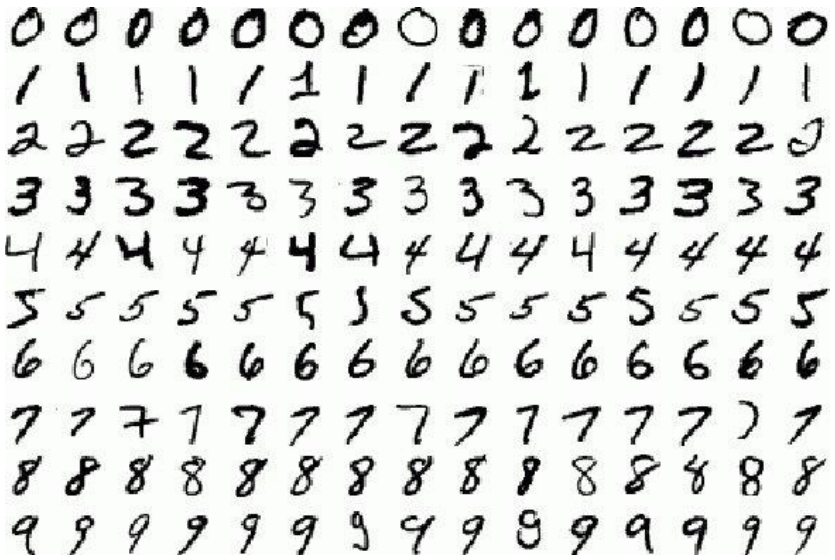
Motivação





- Classificação de emails entre SPAM e pessoal (HAM).
- Detecção de símbolos (classificação de símbolos).

Motivação



- Reconhecimento de dígitos escritos à mão.
- Classificação de texto.

Definição do problema

Problema: atribuir a cada exemplo de entrada o ***rótulo*** correspondente a uma das Q classes existentes, $C_q, q = 1, \dots Q$, à qual o exemplo pertence.

- Este tipo de desafio é característico de problemas conhecidos como ***classificação***.

- Semelhante ao problema da regressão linear, existe um conjunto de treinamento $\{\boldsymbol{x}(i); y(i)\}_{i=0}^{N-1}$ que é utilizado para treinar um ***classificador***, onde
  - $\boldsymbol{x}(i) = [x_1(i) \quad \cdots \quad x_K(i)]^T \in \mathbb{R}^{K\times 1}$ representa o *i*-ésimo vetor exemplo de entrada, o qual é caraterizado por K atributos, $x_1, \dots, x_K$
  - e $y(i) \in \mathbb{R}$ representa o *i*-ésimo ***rótulo***. Como veremos à seguir, y pode ser um escalar $\mathbb{R}^1$ ou um vetor $\mathbb{R}^{Q\times 1}$.

Representação da saída desejada

- A saída desejada para o exemplo de entrada deve ser o ***rótulo*** da classe à qual ele pertence.
- Sendo assim, a saída y de um ***classificador***, é uma variável categórica (ou seja, discreta).
- Portanto, para realizarmos o treinamento do modelo, é necessário escolher uma ***representação numérica*** para a saída desejada, ou seja, y.
- Assim, como veremos a seguir, duas opções podem ser adotadas, dependendo do tipo de classificação a ser feita.

Representação da saída desejada

- **Classificação binária**: existem apenas duas classes possíveis, C_1 e C_2. Portanto, neste caso, podemos utilizar ***uma única saída escalar binária*** para indicar a classe correspondente ao exemplo de entrada:

$$y(i) = \begin{cases} 0, & \boldsymbol{x}(i) \in C_1 \\ 1, & \boldsymbol{x}(i) \in C_2 \end{cases}$$

- Assim, $y(i) \in \mathbb{R}^1$, de maneira que o classificador realiza um mapeamento $\mathbb{R}^K \rightarrow \mathbb{R}^1$
- Também é possível utilizar $y(i) = -1$ para $\boldsymbol{x}(i) \in C_1$.

Representação da saída desejada

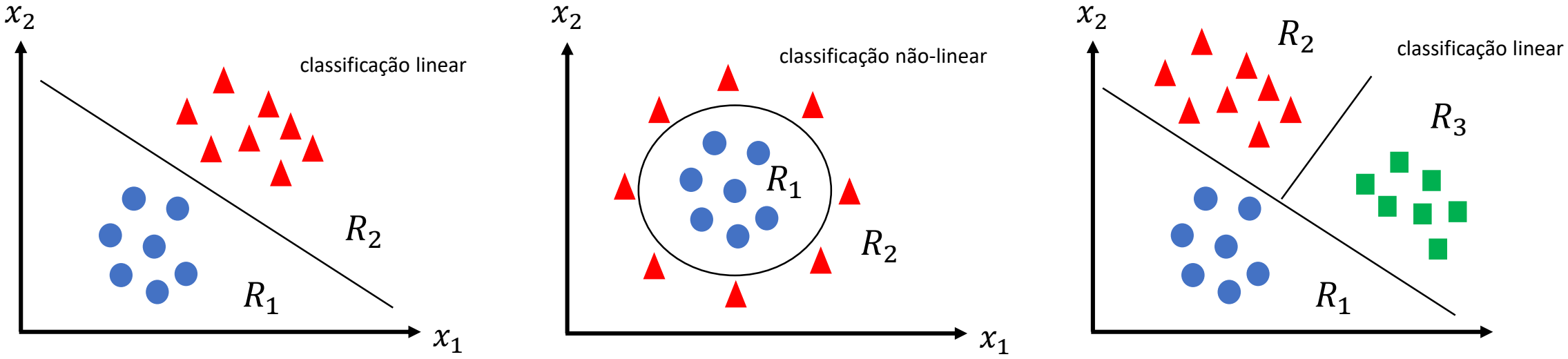
- **Classificação multi-classes**: existem mais de 2 classes possíveis ($Q > 2$).
- Uma estratégia bastante utilizada para representar estas classes é conhecida como ***one-hot encoding***.
- ***one-hot encoding***: utiliza uma representação binária para cada uma das variáveis categóricas.
- Neste caso, o ***classificador*** produz múltiplas saídas, cada uma representando a ***possibilidade*** (ou probabilidade) do padrão pertencer a uma classe específica.
- **Exemplo**: imaginemos um classificador de texto com quatro classes possíveis: *esporte*, *política*, *ciências* e *variedades*. Como vocês as representariam com o ***one-hot encoding***?

$$\left.\begin{array}{ll} \textit{esporte:} & [1 \quad 0 \quad 0 \quad 0]^T \\ \textit{política:} & [0 \quad 1 \quad 0 \quad 0]^T \\ \textit{ciências:} & [0 \quad 0 \quad 1 \quad 0]^T \\ \textit{variedades:} & [0 \quad 0 \quad 0 \quad 1]^T \end{array}\right\}$$

Assim, $\boldsymbol{y}(i) \in \mathbb{R}^{Q \times 1}$, de maneira que o classificador realiza um mapeamento $\mathbb{R}^K \rightarrow \mathbb{R}^Q$.

Fronteiras de decisão de um classificador

- O espaço de entrada $\mathbb{R}^K$ é dividido em ***regiões de decisão*** $R_i, i = 1, \ldots, \mathrm{Q}$, as quais são delimitadas ou separadas pelas ***fonteiras de decisão***, que correspondem a ***superfícies*** (ou ***superfícies de decisão***) no espaço dos atributos onde ocorre uma indeterminação, ou, analogamente, um empate entre diferentes classes possíveis.
- As ***fronteiras de decisão*** podem ser ***lineares*** (e.g., retas e planos) ou ***não-lineares*** (e.g., círculos e esferas).



Regiões de decisão em problemas de classificação binária e multi-classes.

Classificação linear

- Como vimos anteriormente, o objetivo da ***classificação*** é usar as características (i.e., atributos) de, por exemplo, um objeto para identificar a qual classe (ou grupo) ele pertence.
- Um ***classificador linear*** atinge esse objetivo tomando uma decisão de classificação com base no valor de uma ***combinação linear*** dos atributos.
- A saída de um classificador linear é dada por

$$y = f\left(\sum_{k=1}^{K} a_k x_k\right),$$

onde $f(.)$ é uma função que converte o produto escalar dos dois vetores na saída desejada, ou seja, na classe do objeto.

- ***Classificadores lineares*** sãos frequentemente usados em situações em que a velocidade da classificação é um problema, pois ele geralmente é o classificador mais rápido.
- Além disso, os ***classificadores lineares*** geralmente funcionam muito bem quando o número de atributos é grande, como no caso da classificação de documentos.

Teoria bayesiana de decisão

- A teoria bayesiana de decisão é uma ***abordagem estatística*** para o problema de ***classificação***.
- Ela explora o conhecimento de probabilidades ligadas às classes e aos atributos, bem como dos ***custos*** associados a cada decisão, para realizar a classificação de cada novo exemplo.
- **Definições**: Considere que um exemplo (conjunto de atributos) a ser classificado seja descrito por um vetor de atributos $\boldsymbol{x} \in \mathbb{R}^{K\times 1}$. Cada exemplo pertence a uma, e somente uma, classe C_q, sendo que existem ao todo Q classes possíveis.
  - $P(C_q)$ denota a probabilidade ***a priori*** associada à classe C_q.
  - Em outras palavras, $P(C_q)$ indica a probabilidade de um exemplo arbitrário (e desconhecido) pertencer à classe C_q.

Teoria Bayesiana de decisão

- Agora suponha, que um exemplo $\boldsymbol{x}$ seja observado. De posse do conhecimento das características deste exemplo, qual deve ser a decisão quanto à classe a que ele pertence?
  - Uma opção intuitiva e bastante poderosa é escolher a classe que se mostre a mais provável tendo em vista os atributos específicos do exemplo, $\boldsymbol{x}$.
  - Ou seja, a decisão é tomada em favor da classe cuja probabilidade ***a posteriori*** (ou seja, já levando em consideração o conhecimento do vetor de atributos) seja máxima.
  - A probabilidade ***a posteriori*** corresponde à probabilidade condicional $P(C_q|\boldsymbol{x})$.
  - Como calculamos $P(C_q|\boldsymbol{x})$?
- **Teorema de Bayes**

$$P(C_q|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_q)P(C_q)}{P(\boldsymbol{x})}$$

onde o termo $P(\boldsymbol{x}|C_q)$ é denominado de ***verossimilhança*** (***likelihood***) e o termo $P(\boldsymbol{x})$ é normalmente chamado de ***evidência***.

Máxima probabilidade a posteriori (MAP)

- A opção intuitiva sugerida anteriormente é conhecida como o critério ou decisor da ***máxima probabilidade a posteriori*** (MAP, do inglês ***maximum a posteriori probability***), cuja decisão para o padrão $\boldsymbol{x}$ é dada pela classe C_q que maximiza $P(C_i|\boldsymbol{x})$, ou seja, em forma matemática:

$$\text{MAP: } C_q = \arg \max_{C_i, i=1,\ldots,Q} P(C_i|\boldsymbol{x}).$$

Probabilidade a posteriori

- Observe que, com base no teorema de Bayes, a solução para a equação acima é equivalente àquela que maximiza o numerador, $P(\boldsymbol{x}|C_i)P(C_i)$, de forma que:

$$\text{MAP: } C_q = \arg \max_{C_i, i=1,\ldots,Q} P(\boldsymbol{x}|C_i)P(C_i),$$

já que o denominador $P(\boldsymbol{x})$ não depende das classes testadas, servindo apenas como fator de escala no critério.

Máxima verossimilhança (ML)

- O ***decisor de máxima verossimilhança*** (ML, do inglês ***maximum likelihood***) parte do pressuposto de que não há informação estatística consistente sobre as classes, i.e., sobre $P(C_i)$.
- Portanto, o critério ML toma a decisão em favor da classe que apresenta o maior valor para a probabilidade $P(\boldsymbol{x}|C_i)$. Neste sentido, o ML escolhe a classe C_q mais plausível ou mais verossímil em relação ao padrão observado:

$$\text{ML: } C_q = \arg \max_{C_i, i=1,\ldots,Q} P(\boldsymbol{x}|C_i)$$

- **OBS**.: se compararmos as expressões associadas aos critérios MAP e ML, percebemos que a diferença fundamental entre eles reside no fato de o MAP explicitamente incorporar o conhecimento das ***probabilidades a priori***, i.e., $P(C_i)$. Curiosamente, quando temos um cenário em que as classes são equiprováveis, i.e., $P(C_i) = 1/Q$ e independentes do índice, i. Então, maximizar a ***probabilidade a posteriori*** fornecerá a mesma solução que o ML.

Exemplo: diagnóstico de doenças

Vamos supor que estamos trabalhando no diagnóstico de uma nova doença, e que fizemos testes em 100 indivíduos distintos. Após coletarmos os resultados, descobrimos que 20 deles possuíam a doença (20%) e 80 estavam saudáveis (80%), sendo que dos indivíduos que possuíam a doença, 90% receberam positivo no teste da doença, e 30% deles que não possuíam a doença também receberam o teste positivo.

- **Pergunta**: Se uma novo indivíduo realizar o teste e receber um resultado positivo, qual a probabilidade de ele realmente possuir a doença?

Exemplo: diagnóstico de doenças (Solução)

Informações que possuímos:

2 classes: possui doença e não possui doença	1 atributo: resultado do teste: + ou –

- Pergunta em forma probabilistica: $P(\text{doença}|+)$, ou seja, probabilidade do indivíduo ter a doença dado que o resultado observado é positivo?
- Probabilidades:

$P(+\|\text{doença}) = 0.9$	$P(+\|\text{sem\_doença}) = 0.3$
$P(\text{doença}) = 0.2$	$P(\text{sem\_doença}) = 0.8$
$P(+) = P(+\|\text{doença})P(\text{doença}) + P(+\|\text{sem\_doença})\ P(\text{sem\_doença}) = 0.42$	

- Usando o teorema de Bayes

$$P(\text{doença}|+) = \frac{P(+|\text{doença})P(\text{doença})}{P(+)} = 0.429$$

$$P(\text{sem\_doença}|+) = \frac{P(+|\text{sem\_doença})P(\text{sem\_doença})}{P(+)} = 0.571$$

A probabilidade dele não ter a doença mesmo tendo seu teste positivo é de aproximadamente 57%, ou seja, a probabilidade de ***falsos positivos*** é alta. Portanto, este não é um teste confiável.

Classificador naïve Bayes

- São classificadores que assumem que os ***atributos*** são ***estatisticamente independentes*** uns dos outros.
- Ou seja, a alteração do valor de um atributo, não influencia diretamente ou altera o valor de qualquer um dos outros atributos.
- Assim a probabilidade da classe C_q dado o vetor de atributos $\boldsymbol{x}$ pode ser reescrita como

$$P\left(C_q|\boldsymbol{x} = [x_1 \quad \cdots \quad x_K]^T\right) = \frac{P\left(\boldsymbol{x}|C_q\right)P\left(C_q\right)}{P(\boldsymbol{x})} = \frac{P\left(x_1|C_q\right)\ldots P\left(x_K|C_q\right)P\left(C_q\right)}{P(x_1)\ldots P(x_K)}$$

- Com a independência dos atributos, os decisores MAP e ML são dados por

$$\text{MAP: } C_q = \arg\max_{C_i, i=1,\ldots,Q} P(x_1|C_i)\ldots P(x_K|C_i)P(C_i),$$

$$\text{ML: } C_q = \arg\max_{C_i, i=1,\ldots,Q} P(x_1|C_i)\ldots P(x_K|C_i).$$

- Aplicações típicas do classificador naive Bayes incluem filtragem de spam, classificação de documentos e previsão de sentimentos.

Classificador naïve Bayes

Vantagens

- Fácil de ser implementado e altamente escalável.
- Funciona bem mesmo com poucos dados.
- Rápido para realizar as classificações, e portanto, pode ser utilizado em aplicações de tempo-real.
- Além de simples, ele é conhecido por apresentar performance melhor do que métodos de classificação altamente sofisticados em algumas aplicações.

Desvantagens

- Assume que todos os atributos são independentes, o que muitas vezes não é verdade na prática.
- Não consegue classificar caso uma das probabilidades condicionais seja igual a zero, mas existem formas de se driblar esse problema (e.g., técnica da suavização de Laplace).
- É necessário se conhecer ou se assumir as probabilidades condicionais dos atributos.

Tipos de classificadores naïve Bayes

- Na prática, as probabilidades condicionais dos atributos x_k de uma classe, C_q, $P(x_k|C_q), \forall k$, são geralmente modeladas usando-se o mesmo tipo de distribuição de probabilidade, como as distribuições Gaussiana, Multinominal e de Bernoulli.
- Portando, tem-se 3 tipos diferentes de classificadores dependendo da suposição feita para a probabilidade condicional $P(x_k|C_q)$:
  - Classificador naïve Bayes Gaussiano
  - Classificador naïve Bayes Multinomial
  - Classificador naïve Bayes Bernoulli

Classificador naïve Bayes Gaussiano

- Quando lidamos com ***atributos*** x_1, ..., x_K, que apresentam ***valores contínuos***, uma suposição típica é que os valores dos atributos sejam distribuídos de acordo com uma ***distribuição normal*** (ou **Gaussiana**).
- Para se encontrar os parâmetros do classificador faz-se o seguinte:
  - Primeiro, segmenta-se os atributos, x_1, ..., x_K, de acordo com a classe a que pertencem;
  - Em seguida, calcula-se a média, μ_{x_k,C_q}, e a variância, $\sigma^2_{x_k,C_q}$, de cada atributo x_k em relação à classe, C_q, a que pertence.
- Assim, a probabilidade condicional $P(x_k|C_q)$ pode ser calculada inserindo-se o valor de x_k na equação da distribuição Normal parametrizada com μ_{x_k,C_q} e $\sigma^2_{x_k,C_q}$.

$$P(x_k|C_q) = \frac{1}{\sigma^2_{x_k,C_q}\sqrt{2\pi}} e^{-(x_k-\mu_{x_k,C_q})^2 \Big/ 2\sigma^2_{x_k,C_q}}.$$

- Essa é outra suposição forte, pois muitos atributos não seguem uma distribuição normal. Embora isso seja verdade, supondo uma distribuição normal torna os cálculos muito mais fáceis.

Exemplo: Probabilidade da prática de esportes

Nesse exemplo vamos usar o classficador Naive Bayes Gaussiano para calcular a probabilidade dos jogadores jogarem ou não, com base nas condições climáticas. Baseado nos dados abaixo, qual a probabilidade dos jogadores jogarem se temperatura = 25 °C e humidade = 62%?

Temperatura [°C]	Humidade [%]	Jogar?
29.44	85	Não
26.67	90	Não
28.33	86	Sim
21.11	96	Sim
20.00	80	Sim
18.33	70	Não
17.78	65	Sim
22.22	95	Não
20.56	70	Sim
23.89	80	Sim
23.89	70	Sim
22.22	90	Sim
27.22	75	Sim
21.67	91	Não

Exemplo: Probabilidade da prática de esportes

Primeiro, precisamos calcular a média e variância para cada atributo, ou seja, para temperatura e humidade.

Temperatura [°C]	
E[temp. | jogar=sim]	22.78
std(temp. | jogar=sim)	3.42
E[temp. | jogar=não]	23.67
std(temp. | jogar=não)	4.39

Humidade [%]	
E[hum. | jogar=sim]	79.11
std(hum. | jogar=sim)	10.22
E[hum. | jogar=não]	86.20
std(hum. | jogar=não)	9.73

P(jogar=sim)	9/14
P(jogar=não)	5/14

$P(\text{temp.}=25 \mid \text{jogar=sim}) = \frac{1}{3.42\sqrt{2\pi}} e^{-(25-22.78)^2 / 2(3.42)^2} = 0.0944$

$P(\text{hum.}=62 \mid \text{jogar=sim}) = \frac{1}{10.22\sqrt{2\pi}} e^{-(62-79.11)^2 / 2(10.22)^2} = 0.0096$

$P(\text{temp.}=25 \mid \text{jogar=não}) = \frac{1}{4.39\sqrt{2\pi}} e^{-(25-23.67)^2 / 2(4.39)^2} = 0.0869$

$P(\text{hum.}=62 \mid \text{jogar=não}) = \frac{1}{9.73\sqrt{2\pi}} e^{-(62-86.2)^2 / 2(9.73)^2} = 0.0019$

Agora calculamos as probabilidades:

- *P*(jogar=sim | temp.=25, hum.=62) = *P*(temp.=25 | jogar=sim) *P*(hum.=62 | jogar=sim)*P*(jogar=sim) = 5.83e-4
- *P*(jogar=não | temp.=25, hum.=62) = *P*(temp.=25 | jogar=não) *P*(hum.=62 | jogar=não)*P*(jogar=não) = 5.78e-5

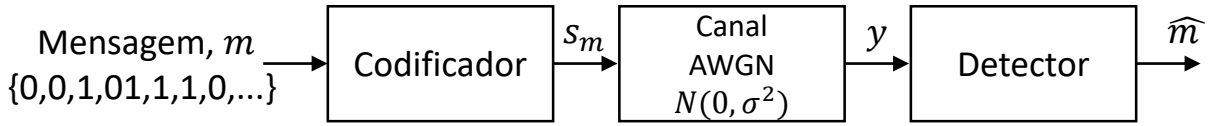
Portanto, a probabilidade é maior para o caso deles jogarem.

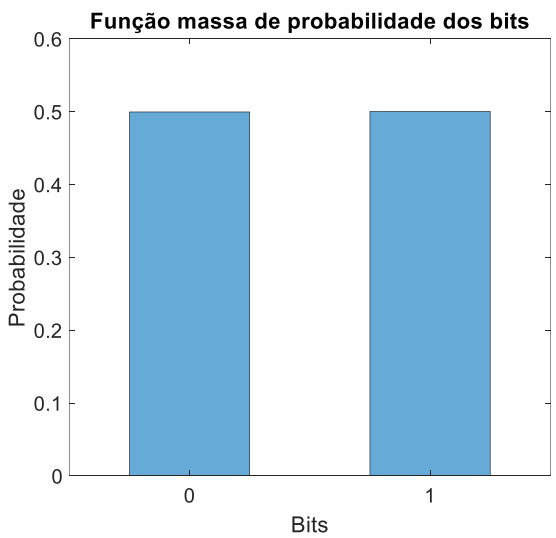
Exemplo: Detecção de símbolos BPSK em canais AWGN

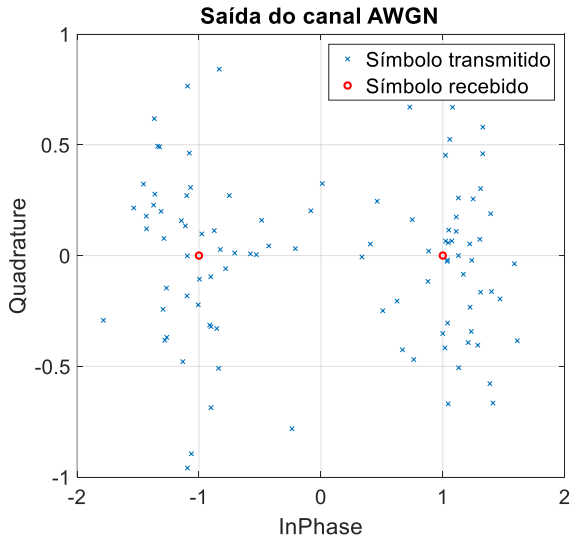
- Imagine um codificador que converte o m-ésimo bit de uma mensagem composta por 0s e 1s nos símbolos s_0 = -1 e s_1 = 1, para $m = 0$ e 1, respectivamente.
- Em seguida, os símbolos codificados passam por um canal AWGN cuja saída é dada por

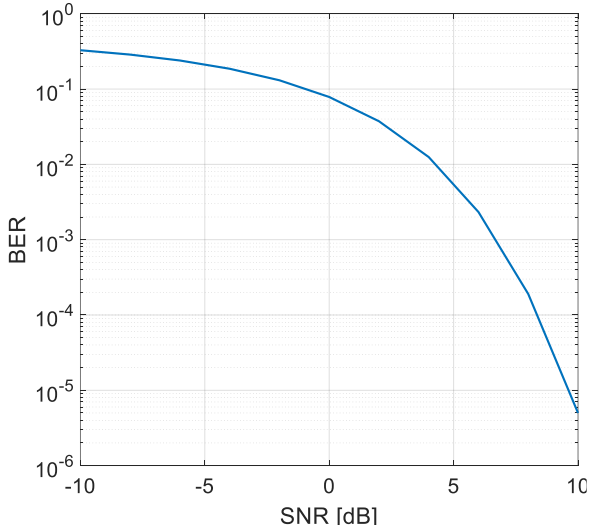
$$y = s_m + w, \text{ onde } w \sim N(0, \sigma^2).$$

- Finalmente, o detector tem a tarefa de recuperar os bits transmitidos de tal forma que a probabilidade de erro, $P_e = P(\hat{m} \neq m)$, seja minimizada.









Exemplo: Detecção de símbolos BPSK em AWGN

- Detector MAP para esse problema é dado por:

$$S_m = \arg \max_{S_m,\, m=0,1} P(S_m|y) = \arg \max_{S_m, m=0,1} P(y|S_m)P(S_m)$$

- Se o símbolo s_0 é transmitido, então o sinal recebido é dado por: $y = s_0 + w$

$$P(y|s_0) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(y+1)^2/2\sigma^2}.$$

- Se o símbolo s_1 é transmitido, então o sinal recebido é dado por: $y = s_1 + w$

$$P(y|s_1) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-1)^2/2\sigma^2}.$$

- Como $P(S_0)$ = $P(S_1)$ = 1/2, então o detector MAP é equivalente ao ML, e assim

$$S_m = \arg \max_{S_m, m=0,1} \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-S_m)^2/2\sigma^2} = \arg \max_{S_m, m=0,1} (y - S_m)^2.$$

Exemplo: Detecção BPSK com Scikit-Learn

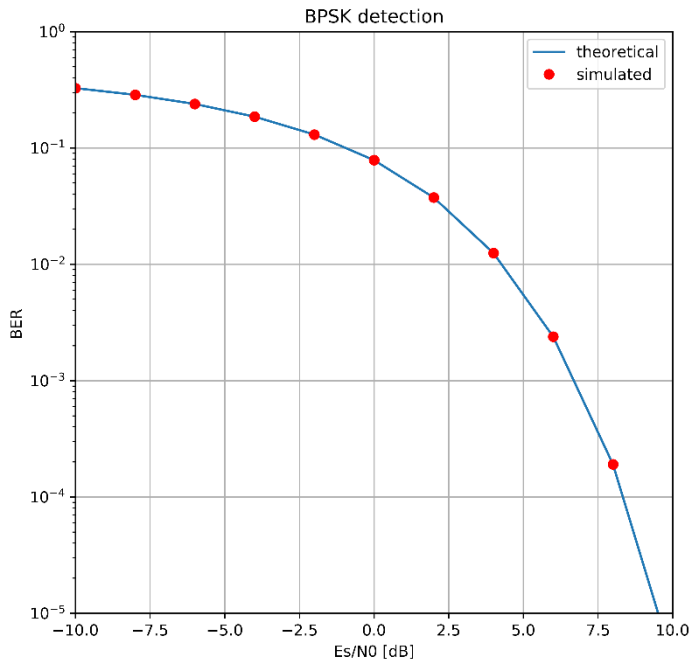
```
# Import all necessary libraries.
import numpy as np
from scipy.special import erfc
from sklearn.naive_bayes import GaussianNB

# Number of BPSK symbols to be transmitted.
N = 1000000
# Instantiate a Gaussian naive Bayes classifier.
gnb = GaussianNB()

# Create Es/N0 vector.
EsN0dB = np.arange(-10,12,2)
ber_theo = ber_simu = np.zeros(len(EsN0dB))
for idx in range(0,len(EsN0dB)):
    EsN0Lin = 10.0**(-(EsN0dB[idx]/10.0))
    # Generate N BPSK symbols.
    x = (2.0 * (np.random.rand(N) >= 0.5) - 1.0).reshape(N, 1)
    # Generate noise vector
    noise = np.sqrt(EsN0Lin/2.0)*np.random.randn(N, 1)
    # Pass symbols through AWGN channel.
    y = x + noise
    # Fit.
    gnb.fit(y, x.ravel())
    # Predict.
    detected_x = gnb.predict(y).reshape(N, 1)
    # Simulated BPSK BER.
    ber_simu[idx] = 1.0 * ((x != detected_x).sum()) / N
    # Theoretical BPSK BER.
    ber_theo[idx] = 0.5*erfc(np.sqrt(10.0**((EsN0dB[idx]/10.0))))
```

Importa classe GaussianNB do módulo naive_bayes do biblioteca SciKit-Learn.

Instancia objeto da classe GaussianNB.

Treina o classificador.

Executa a classificação.



Exemplo: bpsk_detection.ipynb

- Curva simulada se aproxima da teórica.

Classificador naïve Bayes Multinomial

- Com um classificador naïve Bayes multinomial, os ***atributos*** são ***discretos*** e representam as frequências com as quais determinados eventos são gerados por uma distribuição multinomial, com probabilidades $(p_1, p_2, \dots, p_K)$, onde p_k é a probabilidade de que o evento k ocorra.
- Desta forma, o vetor de atributos $\boldsymbol{x} = [x_1, x_2, \dots, x_K]^T$ é então, um ***histograma***, com x_k contando o número de vezes que o evento k foi observado em uma instância específica.
- Este classificador é normalmente usado para classificação de documentos, com eventos representando a ocorrência de uma palavra no documento.
- A probabilidade de observar um histograma $\boldsymbol{x}$ é dada por

$$P(\boldsymbol{x}|C_q) = \frac{(\sum_k x_k)!}{\prod_k x_k!} \prod_k {p_{qk}}^{x_k},$$

onde p_{qk} é a probabilidade da classe C_q gerar o atributo x_k.

Exemplo: classificador multinomial com Scikit-Learn

```
# Import all necessary libraries.
from sklearn.datasets import fetch_20newsgroups
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix

# Use the "20 Newsgroups corpus" from scikit to show how we might classify these
short documents into categories.
data = fetch_20newsgroups()
data.target_names

# Select just a few of these categories, and download the training and testing set.
categories = ['talk.religion.misc', 'soc.religion.christian', 'sci.space', 'comp.graphics']
train = fetch_20newsgroups(subset='train', categories=categories)
test = fetch_20newsgroups(subset='test', categories=categories)

# Convert a collection of text documents to a matrix of token counts.
cv = CountVectorizer()
# Naive Bayes classifier for multinomial models.
mnb = MultinomialNB()
# Create a pipeline that attaches the vectorizer to a multinomial naive Bayes
classifier.
model = make_pipeline(cv, mnb)

# Train model. Apply the model to the training data.
model.fit(train.data, train.target)
# Run validation. Predict labels for the test data.
labels = model.predict(test.data)
```

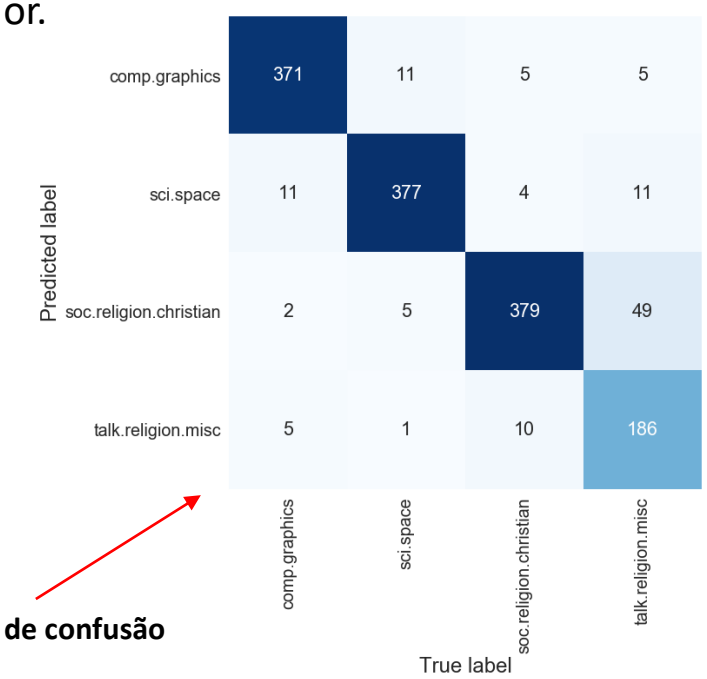
Base com 20 tópicos diferentes de discussão.

Treinamos e validamos com apenas 4 tópicos.

Converte o texto em um conjunto de valores numéricos representativos, ou seja, uma matriz com o número de ocorrências de cada palavra.

Treinamento e validação do classificador.

- Classificação de textos em categorias/classes.
- Esse exemplo usa uma base de dados de grupos de discussão disponibilizada pela biblioteca Scikit-learn.
- Ele classifica textos em 4 classes: ‘religião’, ‘cristianismo’, ‘espaço’ e ‘computadores’.
- O objeto da classe ***CountVectorizer*** cria uma matriz registrando o número de vezes que cada palavra aparece.
- A ***matriz de confusão*** é usada para verificar a performance do classificador.



Matriz de confusão

[Exemplo: ClassifyingTextMultinomialNB.ipynb]

Classificador naïve Bayes Bernoulli

- Esse classificador é baseado na distribuição de Bernoulli, que é uma ***distribuição binária***.
- Portanto, esse classificador considera que os ***atributos*** são ***variáveis binárias*** (i.e., booleanos) independentes, ou seja, o ***atributo*** pode estar presente (True) ou ausente (False).
- Assim como o classificador multinomial, esse classificador é utilizado para tarefas de classificação de documentos, onde atributos binários da ocorrência de termos são usados em vez da frequências de termos.
- Se x_k é um atributo booleano que expressa a ocorrência ou ausência do *i*-ésimo termo de um vocabulário de termos (i.e., palavras), então a probabilidade de um documento pertencer à classe C_q é dado por

$$P(\boldsymbol{x}|C_q) = \prod_k p_{qk}^{x_k}(1 - p_{qk})^{(1-x_k)},$$

  onde p_{qk} é a probabilidade da classe C_q gerar o termo x_k.
- Esse classificador é bastante utilizado para classificar textos curtos e tem o benefício de classificar explicitamente a ausência de termos.

Exemplo: classificador Bernoulli com Scikit-Learn

```
# Import all necessary libraries.
import pandas as pd
from sklearn.naive_bayes import BernoulliNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

# Read SMS data base with pandas.
url = 'https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv'
sms = pd.read_table(url, header=None, names=['label', 'message'])

# Convert label to a numerical variable
sms['label_num'] = sms.label.map({'ham':0, 'spam':1})

# Create feature and label vectors.
X = sms.message
y = sms.label_num

# Split array into random train and test subsets.
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

# Convert a collection of text documents into a matrix of token counts.
vect = CountVectorizer(binary=True)
# Learn the vocabulary dictionary and return term-document matrix.
X_train_t = vect.fit_transform(X_train)

# Instantiate a Bernoulli Naive Bayes model.
nb = BernoulliNB(binarize=None)
# Train the MultinomialNB model.
nb.fit(X_train_t, y_train)

# Transform document into document-term matrix.
X_test_t = vect.transform(X_test)
# Perform classification on an array of test vectors X_test_t.
y_pred_class = nb.predict(X_test_t)
```

Download da base de dados.

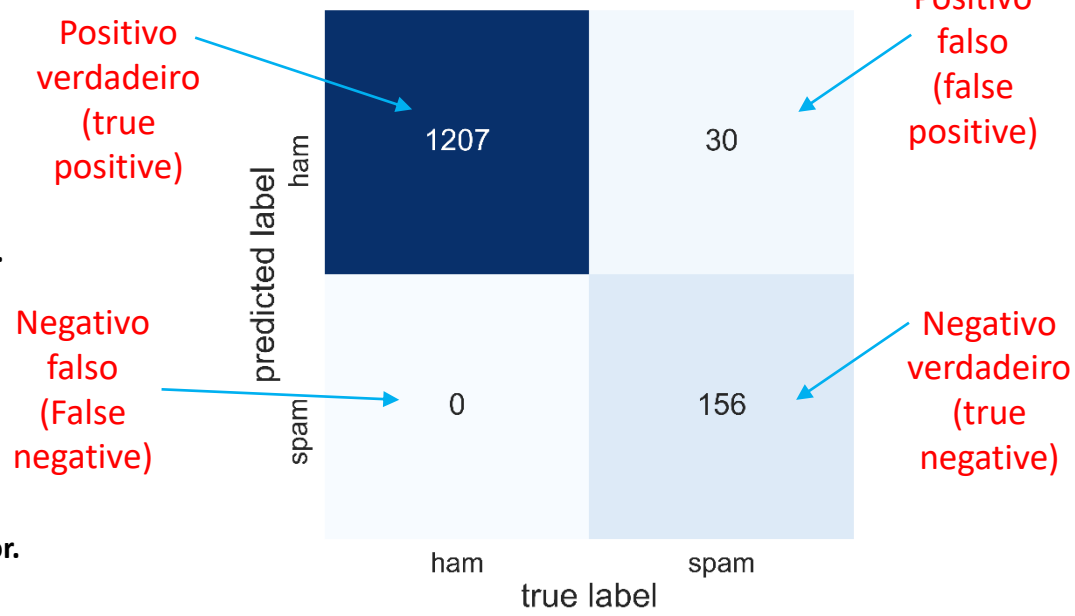
Converte labels em valores discretos.

Divide a base de dados em 75% treinamento e 25% validação.

Cria matriz booleana indicando ou não a presença de uma palavra.

Treinamento e validação do classificador.

- Classificação de mensagens entre SPAM e não-SPAM (HAM).
- Esse exemplo usa uma base de dados de mensagens SMS baixada do GitHub.
- Ele classifica as mensagens em 2 classes: ‘SPAM’ e ‘HAM’.
- O objeto da classe ***CountVectorizer*** cria uma matriz registrando se uma palavra aparece ou não (booleano) em cada mensagem.
- A ***matriz de confusão*** é usada para verificar a performance do classificador.



[Exemplo: SPAMClassificationBernoulliNB.ipynb]

Tarefas e Avisos

- Na próxima aula veremos regressão logística (classificador linear)
- Exemplos vão estar disponíveis no site.
- Lista #4 vai estar no site ainda hoje e já pode ser resolvida.
- Lista #3 pode ser entregue até dia 21/04.
- **IMPORTANTE**: todos os exercícios que envolvam programação devem estar implementados em um notebook do Jupyter. Eu não terei tempo de rodar todos os repos e através do Notebook eu consigo ver os gráficos e resultados.
- Atendimento às quartas-feiras das 8 as 10 da manhã (Skype: zz4fap)
- https://www.inatel.br/docentes/felipefigueiredo/

Obrigado!

Regressão Logística

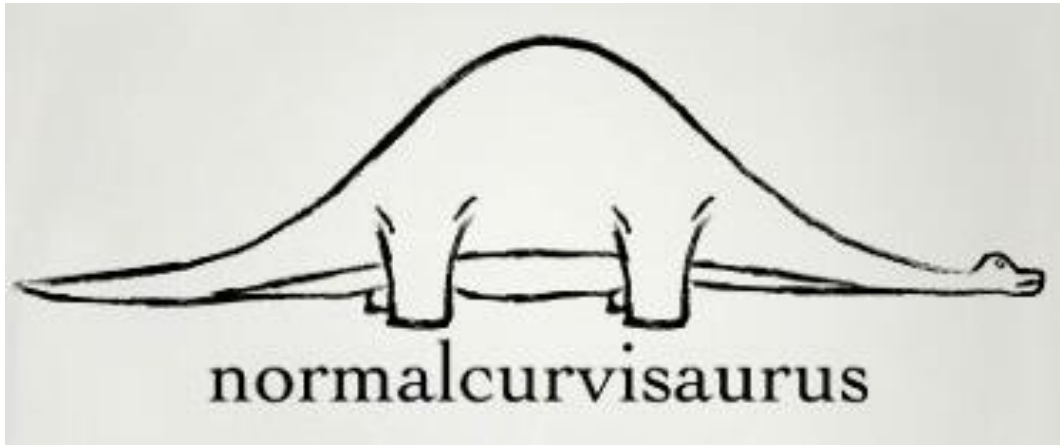
- https://learning.oreilly.com/library/view/hands-on-machine-learning/9781491962282/ch04.html#idm46080001821272
- https://www.cs.cmu.edu/~epxing/Class/10701-10s/Lecture/lecture5.pdf

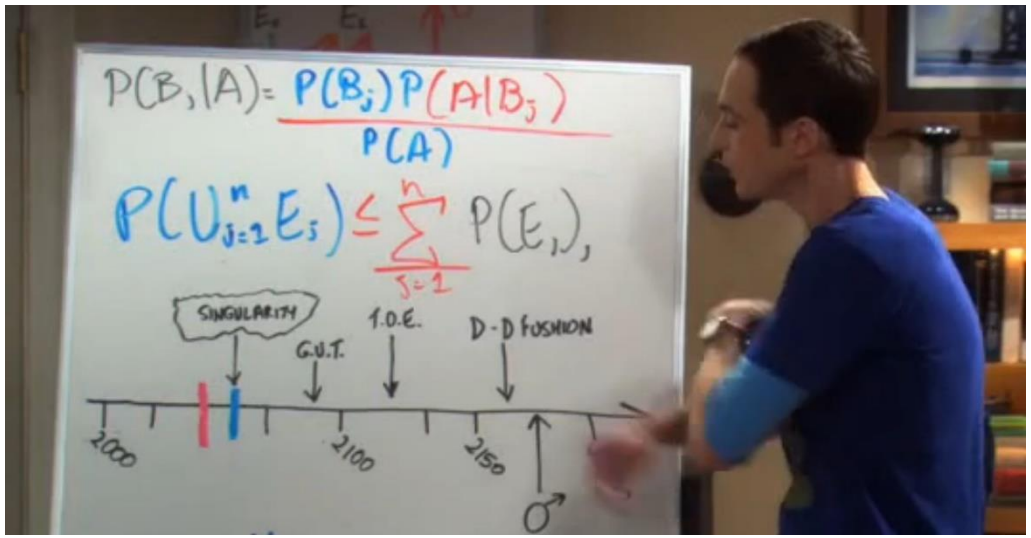
Exemplo: Regressão Logistica

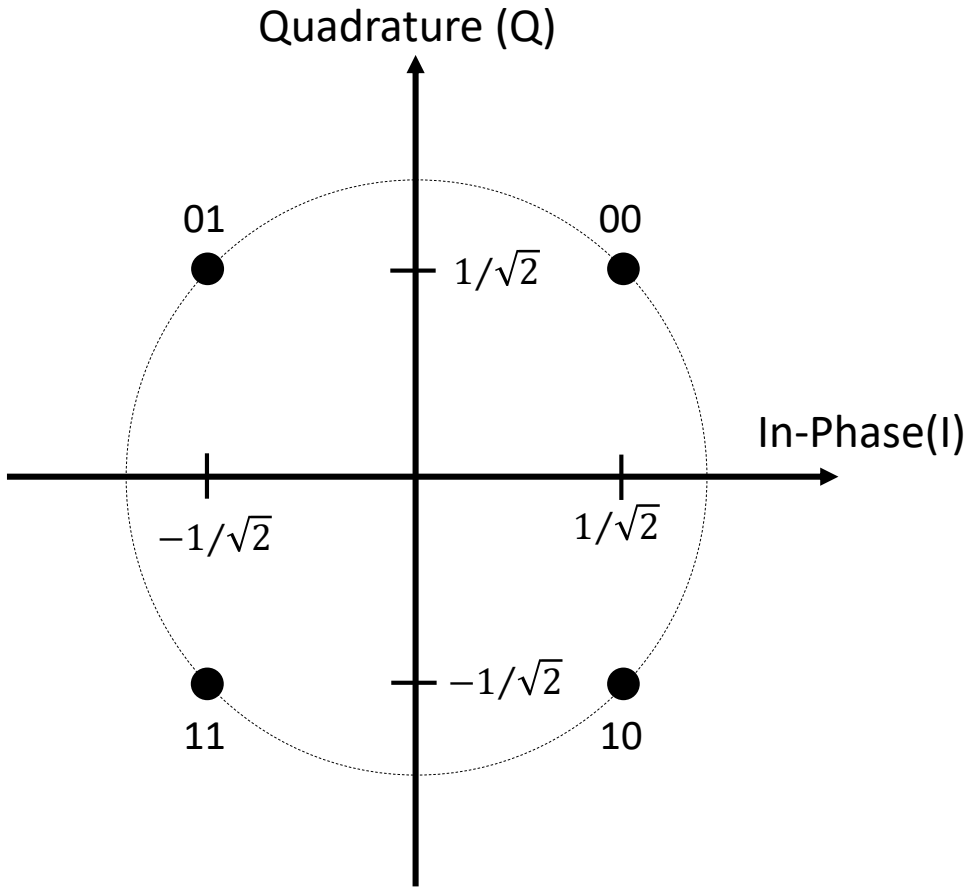
Métricas de avaliação da classificação

Obrigado!


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS


normalcurvisaurus


P(B_i|A) = P(B_j)P(A|B_j) / P(A)
P(U_{j=1}^{n} E_j) ≤ Σ_{j=1}^{n} P(E_j),
SINGULARITY
G.U.T.
T.O.E.
D-D FUSION
2000
2100
2150


Quadrature (Q)
01
00
$1/\sqrt{2}$
In-Phase(I)
$-1/\sqrt{2}$
$1/\sqrt{2}$
$-1/\sqrt{2}$
11
10