

data

Naive Bayes

Bruno Coelho • 03/06/2020
bruno.gomes.coelho@usp.br

Naive Bayes

- Modelo **linear*** apenas de **classificação****
- Derivação estatística interessante

*Na maioria dos casos

**Existem versões para regressão, mas não funcionam bem



Naive Bayes - Tópicos:

- Estimando parâmetros de uma distribuição com MLE/MAP
- Introdução ao Naive Bayes
- Aplicabilidade
- Observações
- Resumo e material extra



Estimando parâmetros de uma distribuição com MLE/MAP



Estimando uma moeda não confiável

- Vamos introduzir alguns conceitos estatísticos antes de chegar no naive bayes;
- Imagine uma moeda não confiável, com uma probabilidade de cara P_h (logo uma probabilidade de coroa $(1-P_t)$)*
- Qual você diria que é a probabilidade de cair cara P_h se:
 - De 100 moedas, 80 caíram cara?

*Do ingles “head” e “tails”

Estimando uma moeda não confiável

- Temos a intuição que deve ser próximo de $P_h = 0.80$ pelo os dados.
- Indiretamente, isso é usar Maximum Likelihood Estimation (MLE) para descobrir a probabilidade P_h ideal.

Se quiser seguir a derivação passo a passo, veja [aqui](#)



Maximum Likelihood Estimation (MLE)

- Podemos modelar o problema utilizando a distribuição de Bernoulli onde o parâmetro θ indica a probabilidade de ser cara P_h
 - Considerando N_h o número de moedas com cara e N_t para coroa, temos um conjunto de jogadas/dataset D
 - A probabilidade de termos visto essas N_h e N_t moedas no dataset D dado o parâmetro θ :

$$P(D | \theta) = \binom{n_H + n_T}{n_H} \theta^{n_H} (1 - \theta)^{n_T},$$

Maximum Likelihood Estimation (MLE)

- Como estamos interessados em estimar o θ , vamos selecionar o θ que maximiza a probabilidade dos dados que vimos:

- $\theta = \operatorname{argmax}(P(D | \theta)) = \operatorname{argmax} \log P(D | \theta)$

- Aplicamos o log para transformar multiplicação em soma¹

- Derivando e igualando a zero (e verificando que a 2a derivada é negativa, logo é um máximo) chegamos em: $\theta = \frac{n_H}{n_H + n_T}$

- Que é exatamente o que a nossa intuição nos diz!

Problema

- Se em vez de 100 moedas, eu falar que:
 - Joguei 1 moeda e foi coroa, qual a P_h ?
 - Joguei 3 moeda e 2 foram coroa, qual a P_h ?

Problema

- Se em vez de 100 moedas, eu falar que:
 - Joguei 1 moeda e foi coroa, qual a P_h ? **0%**?
 - Joguei 3 moeda e 2 foram coroa, qual a P_h ? **66%**?
- Em ambos os casos, nossa intuição nos diz que provavelmente temos pouca informação para generalizar!

Breve intro a Prior/Posterior

- Podemos tentar lidar com isso falando que vamos assumir que a moeda seja justa (informação prévia, “prior”), e atualizar essa ideia conforme obtemos novos dados (posterior)
- Ou seja, vamos supor que tivermos $(\alpha-1)$ caras e $(\beta-1)^*$ coroas antes de ver nossos dados; Nossa nova estimativa agora é:

$$\hat{\theta} = \frac{n_H + \alpha - 1}{n_H + n_T + \beta + \alpha - 2}$$

*O -1 é devido a como os estatísticos definem



O que isso tem a ver com Naive Bayes?

- O Naive Bayes vai funcionar de uma maneira bem parecida:
- Vai assumir uma distribuição para as nossas features
 - Por exemplo, suponha que os dados sejam uma gaussiana...
- Com essa distribuição, vai tentar achar os parâmetros θ ótimos
 - Na gaussiana, θ representa a média e desvio padrão que se encaixam com os nossos dados
 - Que é justamente a ideia de selecionar o argmax de θ !



Introdução ao Naive Bayes



Construindo o problema

- Vamos nos preocupar em calcular $P(Y=y | X=x)$
 - Isto é, qual a probabilidade de ser da classe y, dado que vimos as features x
 - Por exemplo, num problema entre cachorros e gatos:
 - Qual a probabilidade de ser um cachorro dado que tem 40 cm e 50kg?
 - Qual a probabilidade de ser um gato dado que tem 40 cm e 50kg?
 - Escolhemos a classe que tem maior probabilidade.



Construindo o problema

- Para calcular, utilizamos o Teorema de Bayes para probabilidade condicional:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}.$$

- Calculamos isso para $y=0$ (classe gato por exemplo) e $y=1$ (cachorro)
 - Escolhemos a classe com maior probabilidade
 - Como para ambas as classes vai ter o mesmo denominador $P(X)$, não precisamos considerar ele na conta;
 - $P(y)$ é simplesmente a proporção de cada classe no nosso dataset
 - E $P(\mathbf{x} | y)$?

Calculando $P(x | y)$

- Uma maneira é contar quantas vezes o x aparece nos dados que temos;
- Vamos ver um exemplo com duas classes e duas features, onde queremos classificar o ponto $x=(1, 7)$:

Exemplo: Calculando $P(x | y)$

- Como classificar $x=(1, 7)$:
 - $P(X=(1, 7) | y=1) = 2/5 = 0.4$
 - $P(X=(1, 7) | y=0) = 0/2 = 0$
- Logo classificamos como classe 1.

Feature1	Feature2	Y
1	7	1
1	6	1
2	7	1
1	7	1
2	8	1
1	4	0
2	8	0

Calculando $P(x | y)$

- Problema: Temos dados limitados e com dimensões altas;
- Isso significa que na prática, não haverá dados na sua base com um X específico que você quer saber a classe.
- Por exemplo, o ponto $x=(2, 6)$ não aparece nenhuma vez para nenhuma classe nos nossos dados, então como classificar?

Assumindo independência

- Agora vem o truque do Naive Bayes: Suponha que as suas **features são independentes!**
 - Isso é quase sempre um **pressuposto inválido!**
 - Por exemplo, a altura de um animal e seu peso estão correlacionados - Não são independentes
 - Na prática, para muitos casos o algoritmo funciona bem mesmo se não houver independência.



Assumindo independência

- Calculando $P(\mathbf{x}|y)$ assumindo independência entre as features:

$$P(\mathbf{x}|y) = \prod_{j=1}^{\text{features}} P(x_j|y), \text{ onde } x_j \text{ é o valor da feature j do ponto que temos interesse}$$

Vamos voltar ao exemplo com o ponto $(2, 6)$ que tinha nos dado problema:

Exemplo: Calculando $P(x | y)$ com independência

- $P(X=(2, 6) | y=1) =$
 $= P(X_1 = 2 | y=1) * P(X_2 = 6 | y=1)$
 $= \frac{2}{5} * \frac{1}{5} = 2/25 = 0.08$

Para $y=0$, temos $P = 0$, então novamente escolhemos a classe 1.

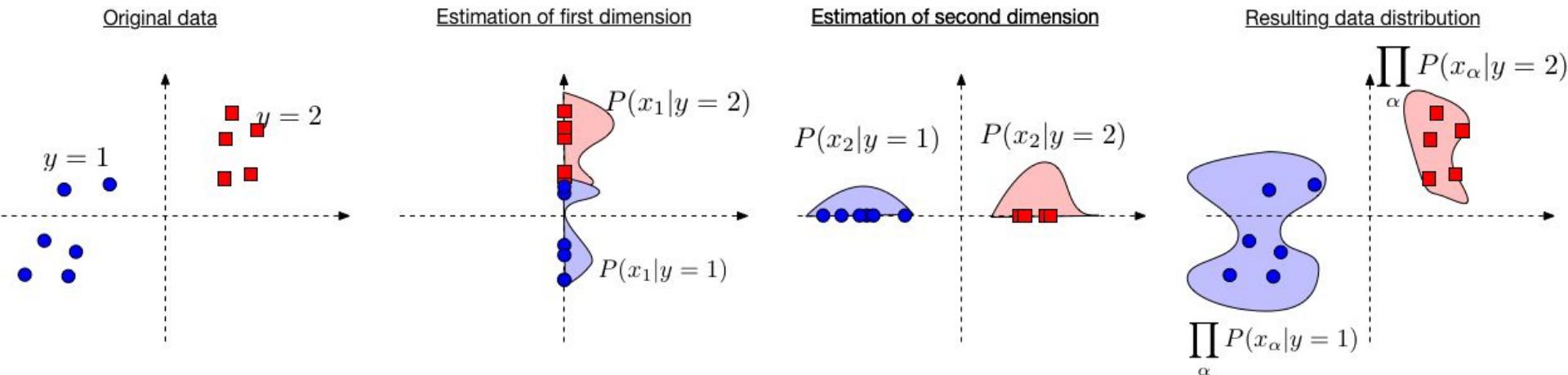
Feature1	Feature2	Y
1	7	1
1	6	1
2	7	1
1	7	1
2	8	1
1	4	0
2	8	0

Calculando $P(x | y)$: Continuação

- Embora assumir independência entre as features já ajuda bastante, não faz muito sentido contar os exemplos que temos acesso uma vez que não temos infinitos dados; Em vez disso:
 - Vamos **supor uma distribuição** para cada feature;
 - Calcular quais são os **parâmetros** dessa distribuição que mais “batem” como nossos dados
 - Usar esses parâmetros para calcular as probabilidades de um novo exemplo para cada feature;
 - Multiplicar tudo e escolher a classe com maior probabilidade.
- Graficamente:



Calculando $P(x | y)$: Continuação



*O exemplo usa classe 1 e 2 em vez de 0 e 1, mas tanto faz
[Fonte](#)

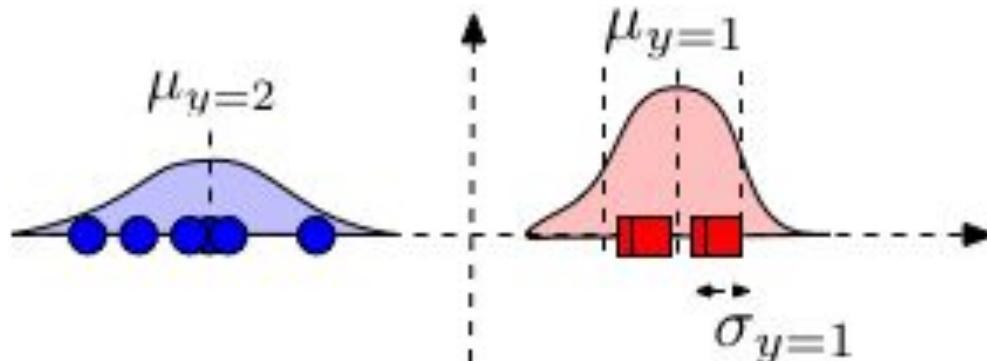


Distribuições usadas

- Dependendo dos dados que temos naquela coluna, selecionamos uma possível distribuição diferente:
 - Dados [binários](#)
 - Dados [categóricos](#)
 - Dados contínuos ([Gaussiana](#))
 - Dados de contagem ([Multinomial](#))

Exemplo: Dados contínuos

- Estimando dados contínuos com uma gaussiana, se resume a calcular os parâmetros de média (μ) e desvio padrão(σ) para cada feature;
- Pensando em 1 feature só:



[Fonte](#)



Aplicabilidade



Aplicabilidade

- Não é necessário padronizar as nossas features uma vez que lidamos com cada uma independentemente;
- Embora estamos assumindo que as nossas features são independentes, mesmo se isso não for verdade, podemos ter um bom classificador
 - Classificação de texto

Aplicabilidade: Classificação de textos

- Em um problema de classificação de texto, podemos classificar se uma determinada fala ou não sobre esportes;
- Convertemos nosso problema para um problema tabulado, contando a quantidade de palavras por frase:
 - Cada linha representa uma frase e a feature é quantas vezes aquela palavra aparece no texto

Aplicabilidade: Classificação de textos

- Frases:

- Frase 1: O jogador tem que jogar muito bem - sobre esportes
- Frase 2: O Bolsonaro tem muito, muito que sair - não é sobre esportes

O	jogador	jogar	muito	bem	Bolson aro	tem	que	sair
1	1	1	1	1	0	1	1	0
1	0	0	2	0	1	1	1	1

Aplicabilidade: Classificação de textos

- Perceba que as nossas features com certeza **não são independentes**:
 - As palavras que escolhemos possuem correlação entre si
- Mesmo assim, o Naive Bayes é um bom algoritmo baseline para esse tipo de problema!
- Como temos dados de contagem, utilizamos um Naive Bayes Multinomial
- Exemplo usando o Sklearn

Observações



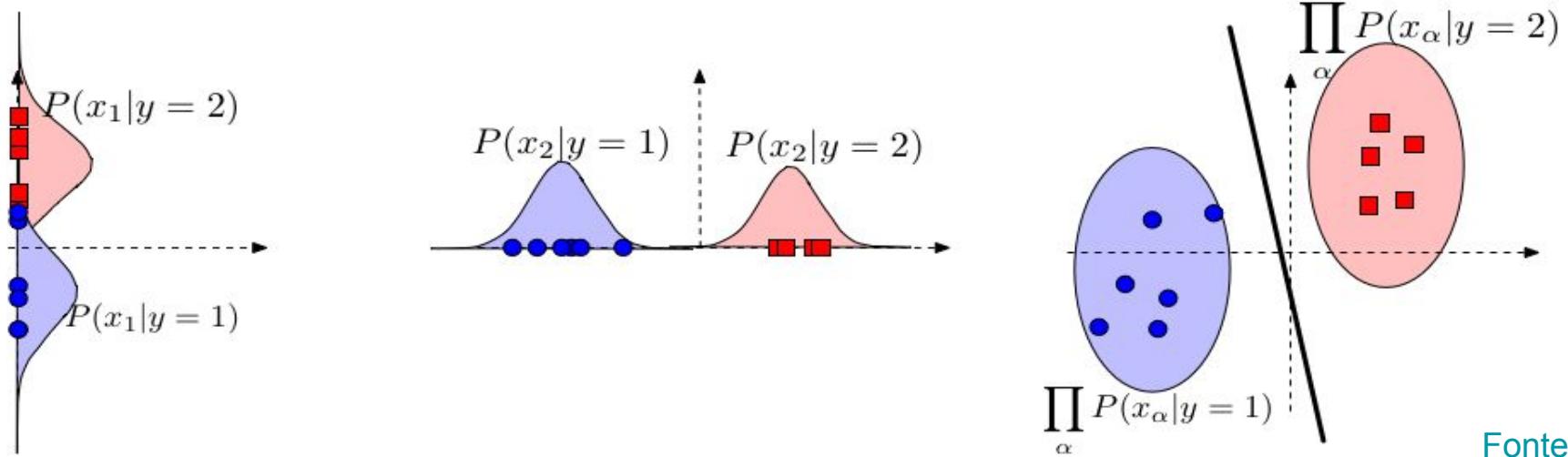
Observações: Probabilidades

- Devido a maneira que o Naive Bayes funciona (usando $P(x|y)$ por trás, chamado também de um modelo gerativo), a probabilidade que ele nos retorna não é interpretável
 - Isto é, se $P_{\text{cachorro}} = 0.8$ (Logo a $P_{\text{gato}} = 0.2$), devemos selecionar a classe cachorro, mas isso não significa que existe 4 vezes mais chance de ser cachorro que gato
- Mais formalmente, as probabilidades não são calibradas



Observações: Linearidade

- Embora não seja intuitivo, um Naive Bayes categórico/gaussiano/multinomial e na grande maioria dos casos, resulta em um classificador **linear**



Observações: Regressão logística

- Se as features são realmente independentes, o Naive Bayes gaussiano tende a exata mesma resposta da regressão logística conforme os dados tendem ao infinito
 - Mas devido a questões de implementação, para um conjunto finito podem ter pequenas divergências.

Observações: Combinando distribuições

- Embora podemos pensar em diferentes colunas com diferentes distribuições (uma gaussiana e outra categórica por exemplo), a maioria das bibliotecas não permite isso e temos que escolher uma distribuição para todos os dados

Resumindo...



Naive bayes

- Modelo de classificação que divide os dados linearmente
- Supõe independência entre as features
- Diferentes versões dependendo de qual distribuição queremos assumir para os dados
- Bom baseline para dados de classificação de texto

Material extra

- [Aula e material](#) do curso maravilhoso de Stanford
 - Derivação passo a passo
- Explicação [interativa](#) do teorema de bayes
- [Introdução visual](#) sobre inferência bayesiana



Fim

