P1 - Statistical Learning

Mateus Marques

19 de outubro de 2022

1 Intro

1.1 Conceitos

- Missing data: é quando uma variável não é observada.
- Missing data at random: algumas variáveis não são observadas em alguns dados de maneira aleatória. Missing data due to systematic reason: existe alguma razão sistemática em que variáveis não são observadas.
- Data prep: é o processo de manipular os dados brutos para que eles estejam prontos para serem analisados. Tarefas usuais são:
  - integração de dados (colocá-los no mesmo formato, unidades etc.).
  - entender os dados (fazer plots, histogramas etc. para ter uma ideia do dataset).
  - discretizar, normalizar os dados.
  - remover outliers (dados muito fora do resto).
  - fazer algo sobre missing data.
  - feature selection.
- Handling missing data:
  - remover linhas ou colunas.
  - imputar a média, mediana ou moda.
  - regressão de outras variáveis.
  - predição, estimação, interpolação.
- Feature: as variáveis que são observadas, correspondem ao $X_1, X_2, \ldots, X_n$.
- Label (rótulo): valores da variáveis *de classe (target)* Y (a que se tem interesse).
- Supervised learning: quando não temos missing labels (dados do Y estão todos presentes).
- Unsupervised learning: quando todo rótulo Y está faltando.
- Semi-supervised learning: alguns labels estão faltando (caso geral).
- Classifier: é uma função $g(X_1, \ldots, X_n)$ que produz labels $\hat{Y}$ em função dos features $X_1, \ldots, X_n$. Produzir uma função g usando os dados é chamado *treinamento do classificador*.

- Classification: é quando Y tem valores discretos.
- Regression: é quando Y tem valores contínuos.
- Training dataset: conjunto de dados utilizados para treinar o classificador (produzi-lo).
- Testing dataset: conjunto de dados utilizados para testar o classificador. Por exemplo estimando a taxa de erro empiricamente.
- Error rate: taxa de erro é a probabilidade do classificador errar $e_g = P(Y \neq g(\mathbf{X}))$.
- Empirical error rate: taxa de erro empírica é a proporção que ele errou nos dados de teste $\hat{e}_g = \frac{\text{número de vezes que } Y \neq g(X)}{\text{número total de tentativas}}$.
- Overfitting: o classificador é muito bom nos dados de teste, mas ele falha para outros dados. Um exemplo é o $1NN$. Nos dados de teste a taxa de erro é sempre zero, mas nos de teste pode ser alta. Outro exemplo é quando o modelo do classificador é muito complexo para o fenômeno, tipo quando se tenta fitar um polinômino de grau 1000 num fenômeno essencialmente linear.
- Underfitting: o classificador treinou pouco demais, o modelo dele é muito simples para conseguir capturar a complexidade do fenômeno. Por exemplo tentar fitar uma reta em um fenômeno altamente não-linear.
- Cross-validation: quando não se tem dados suficiente para criar um *holdout* (separar uma parte dos dados para teste, tipo uns 30%) para testar, então usamos validação cruzada. A ideia é separar uma fração dos dados totais para testar, e ir repetindo sobre toda a base de dados.

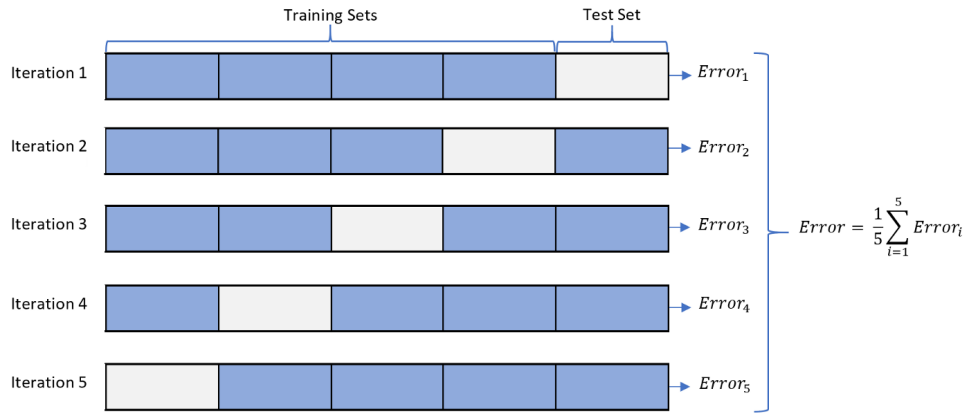

Figura 1: Diagrama de validação cruzada.

- Leave-one-out validation: (é o extremo da validação cruzada) se você tem N linhas de dados, a ideia é fazer validação cruzada com N iterações, sempre deixando um dado para teste de fora em cada separação e depois computando o erro médio sobre as N etapas.

- Validation dataset: parte dos dados que não participam do treino e são utilizados para dar estimativas para a eficiência do classificador e tunar os hiperparâmetros. É diferente do testing dataset, pois o testing dataset é utilizado depois dos hiperparâmetros já terem sido tunados e ele serve para dar uma estimativa imparcial da qualidade do classificador escolhido (através da validação).
- Stratified cross-validation: validação cruzada, porém com uma divisão que respeita a proporção de labels para cada divisão.

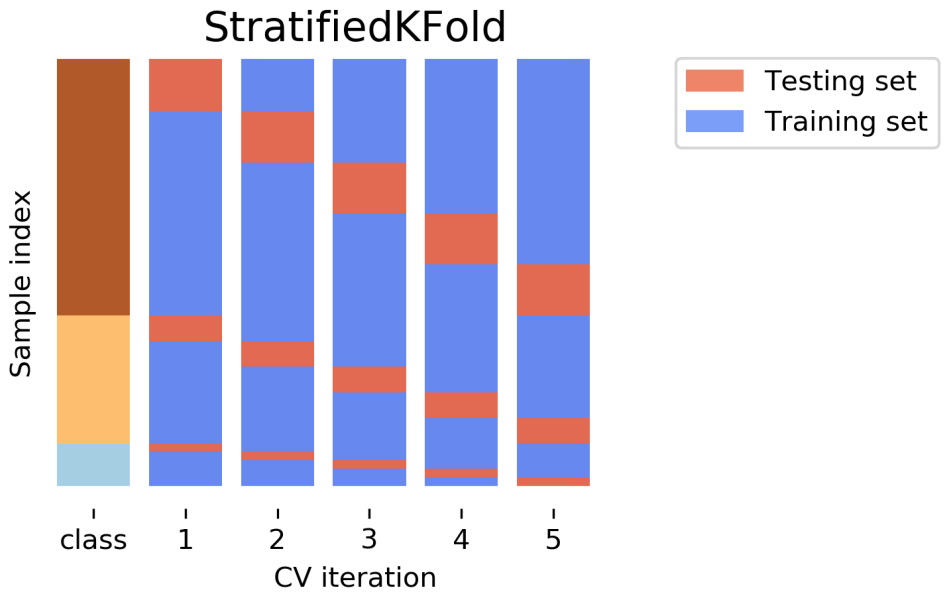

Figura 2: Validação cruzada estratificada.

- Bayes classifier: o classificador ótimo, aquele em que dadas a probabilidade $P(Y = y|X = x)$, ele escolhe o $\hat{Y} = y$ tal que a probabilidade é a maior.
- Bayes error: o menor erro que pode ser cometido por um classificador. É o erro que o classificador de Bayes comete.
- Plug-in classifier: tentar estimar as probabilidades $P(Y = y|X = x)$ através de algum modelo, e então utilizar a fórmula do classificador de Bayes.
- Nearest-neighbor classifier: classificar cada ponto de acordo com uma média entre k vizinhos mais próximos dos dados de treino.
- Teorema sobre kNN: seja n o número de observações no training dataset. Se $k = k(n) \to \infty$ de maneira que $k(n)/n \to 0$, então $\lim_{n\to\infty} E[e_{g_n}] = e_{g^*}$. Ou seja, se pegarmos k tendendo ao infinito mas de maneira que ele seja sempre superado pelo número de observações n, então a taxa de erro do kNN tende para o menor erro possível (erro de Bayes).
- Teorema sobre 1NN: temos para o 1NN que $\lim_{n\to\infty} E[e_{g_n}] \leq 2e_{g^*}$. Ou seja, o erro do 1NN para infinitos dados sempre será menor que duas vezes o erro de Bayes.

1.2 Regression

Assumimos que $Y = f(X) + \epsilon$, onde ϵ é um erro aleatório independente com média zero e variância σ^2. Nós queremos $\hat{Y} = \hat{f}(X)$ para alguma estimativa $\hat{f}$. A medida usual é minimizar o *expected quadratic error* $E[(Y - \hat{Y})^2]$.

Se $\hat{f}$ for muito simples, então caímos em *underfitting*. Se $\hat{f}$ for muito complexo caímos em *overfitting*. Para isso, monitoramos o *mean squared error* com respeito aos dados de TESTE

$$\frac{1}{N}\sum_{i=1}^{N}\Big(y_i - \hat{f}(x_i)\Big)^2,$$

que se trata de uma estimativa para $E[(Y - \hat{Y})^2]$.

Seja D o training dataset. A partir de D, nós treinamos um classificador $\hat{Y} = \hat{f}(X, D)$. O erro quadrático esperado é

$$E[(Y - \hat{Y})^2] = E\Big[E[(Y - \hat{Y})^2|x]\Big].$$

Temos então a seguinte decomposição

$$E[(Y - \hat{Y})^2|x] = E[(f + \epsilon - \hat{f})^2|x] = E\left[\Big((f - E[\hat{f}|x]) + \epsilon + (E[\hat{f}|x] - \hat{f})\Big)^2|x\right] =$$

$$E[(f - E[\hat{f}|x])^2|x] + E[\epsilon^2|x] + E[(E[\hat{f}|x] - \hat{f})^2|x] +$$
$$2E[(f - E[\hat{f}|x])\epsilon|x] + 2E[\epsilon(E[\hat{f}|x] - \hat{f})|x] + 2E[(f - E[\hat{f}|x])(E[\hat{f}|x] - \hat{f})|x]$$

Na expressão acima, cada um dos termos da segunda linha é igual a zero. Pois

- f é "constante" com respeito a $E[\cdot|x]$ e a média de ϵ é nula $E[\epsilon|x] = 0$.

$$2E[(f - E[\hat{f}|x])\epsilon|x] = 2(f - E[\hat{f}|x])\cancelto{0}{E[\epsilon|x]} = 0$$

- A variável ϵ é independente de $\hat{f}$.

$$2E[\epsilon(E[\hat{f}|x] - \hat{f})|x] = 2E[\epsilon|x]\,\cancelto{0}{E[(E[\hat{f}|x] - \hat{f})|x]} = 0.$$

- f é "constante" com respeito a $E[\cdot|x]$.

$$2E[(f - E[\hat{f}|x])(E[\hat{f}|x] - \hat{f})|x] = 2(f - E[\hat{f}|x])\cancelto{0}{E[(E[\hat{f}|x] - \hat{f})|x]} = 0$$

Temos então

$$E[(Y - \hat{Y})^2] = E\Big[E[(Y - \hat{Y})^2|x]\Big] =$$

$$E\Big[E[\epsilon^2|x] + (f - E[\hat{f}|x])^2 + E\big[(E[\hat{f}|x] - \hat{f})^2|x\big]\Big] =$$

$$E[\epsilon^2] + E\Big[(f - E[\hat{f}|x])^2\Big] + E\Big[E\big[(E[\hat{f}|x] - \hat{f})^2|x\big]\Big] \;\Rightarrow$$

$$\boxed{E[(Y - \hat{Y})^2] = \sigma^2 + E[\text{bias}^2] + E[\text{variance of } \hat{f}],}$$

where

$$\text{bias} = f(x) - E[\hat{f}(x, D)|x],$$

$$\text{variance} = E\left[\left(\hat{f} - E[\hat{f}|x]\right)^2 \mid x\right] = E[\hat{f}(x, D)^2|x] - E[\hat{f}(x, D)|x]^2.$$

A ideia intuitiva é

- Um estimador simples geralmente tem vício alto, porém com pouca variância.
- Um estimador complexo geralmente tem vício pequeno, porém variância alta.

2 Evaluation

2.1 Confusion Matrix

A matriz de confusão compara o rótulo de fato (actual label, true class) com o valor dito pelo classificador (predicted class). A primeira letra diz respeito se o actual label é igual ao que foi dito pelo classificador, T = True ou F = False. A segunda letra diz respeito ao valor dito pelo classificador, P = Positive ou N = Negative. Assim temos

$$\begin{pmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{pmatrix},$$

ou a Figura 3 abaixo.

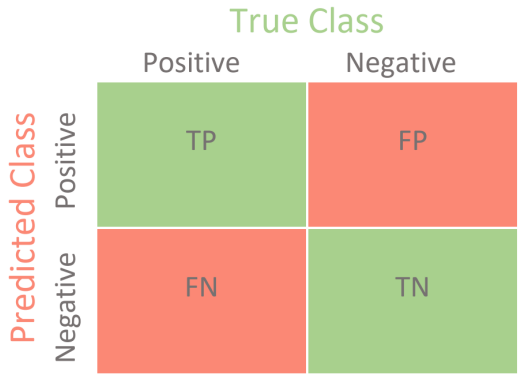

Figura 3: Matriz de confusão.

A acurácia é o quanto o classificador acertou (deu `True`) dentre tudo que ele falou. Ou seja

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} = \frac{\text{true}}{\text{all}}.$$

Alguns problemas com a acurácia são

- No caso de classes desbalanceadas, onde se tem muito mais certo label do que outro. Nesse caso, somente falando o label mais abundante você tem uma acurácia muito alta, porém um classificador burro.

- Alguns erros são piores que outros. No caso da medicina, um falso negativo (não detectar doença existente) é muito pior que um falso positivo (detectar doença inexistente).

A precisão é quanto o classificador acerta quando se tem realmente o label. Ou seja

$$p = \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{true positive}}{\text{positive}} = \frac{\text{true positive}}{\text{get the label}}.$$

Note que no caso de uma doença, um classificador tem alta precisão se ele conseguiu detectar uma fração grande de pessoas que tem doença dentre as que ele falou que tinha (quanto ele acertou sobre quanto ele falou).

A revocação (*recall*) é a TP rate. Ou seja, TP dividido pela soma da coluna. Assim

$$r = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{true positive}}{\text{have the label}}.$$

A ideia da revocação é quanto a taxa que o classificador acerta dentre as pessoas que possuem o label. Isso é uma medida de quanto ele é cuidadoso com o label. Se ele tiver uma revocação alta, quer dizer que dentre as pessoas que tem o label, ele toma um certo cuidado em maximizar o acerto.

Todas as rates é ela mesmo dividida pela soma da coluna. Então, TN rate é $\frac{\text{TN}}{\text{TN+FP}}$.

Sendo p a precisão e r a revocação, F_1 é a média harmônica

$$\frac{2}{F_1} = \frac{1}{p} + \frac{1}{r} \;\Rightarrow\; F_1 = \frac{2pr}{p+r}.$$

Note que sempre $F_1 \leq \min(p, r)$.

2.2 Curva ROCK

Nós só definimos a curva ROC para classificadores que sejam da seguinte maneira:

- Para cada observação x ele calcula um número $h(x)$.
- Então ele classifica $\hat{Y} = 1$ se $h(x) > \alpha$ e $\hat{Y} = 0$ caso contŕario.

A curva ROC (Receiver Operating Characteristic) captura a influêncida do α.

Para classificadores plug-ins que tentam minimizar o custo $c(\hat{Y}, Y)$:

	$Y = 1$	$Y = 0$
$\hat{Y} = 1$	0	a
$\hat{Y} = 0$	b	0

Figura 4: Custo com pesos a e b.

$$E[c(\hat{Y}, Y)] = \sum_x P(X = x) \begin{cases} b\, P(Y = 1|x) \text{ se } g(x) = 0 \\ a\, P(Y = 0|x) \text{ se } g(x) = 1 \end{cases}.$$

Para minimzar, basta tomar

$$\hat{Y} = g(x) = \begin{cases} 1 \text{ se } h(x) = \frac{P(Y=1|X=x)}{P(Y=0|X=x)} > a/b, \\ 0 \text{ caso contrário.} \end{cases}$$

A curva ROC então é o plot de false positive × true positive para vários valores de α.

- (1,1) é o ponto para α muito pequeno.
- (0,0) é o ponto para α muito grande.
- (0,1) é o melhor ponto possível.
- A diagonal de (0,0) até (1,1) é o pior classificador possível. O output é independente de Y.

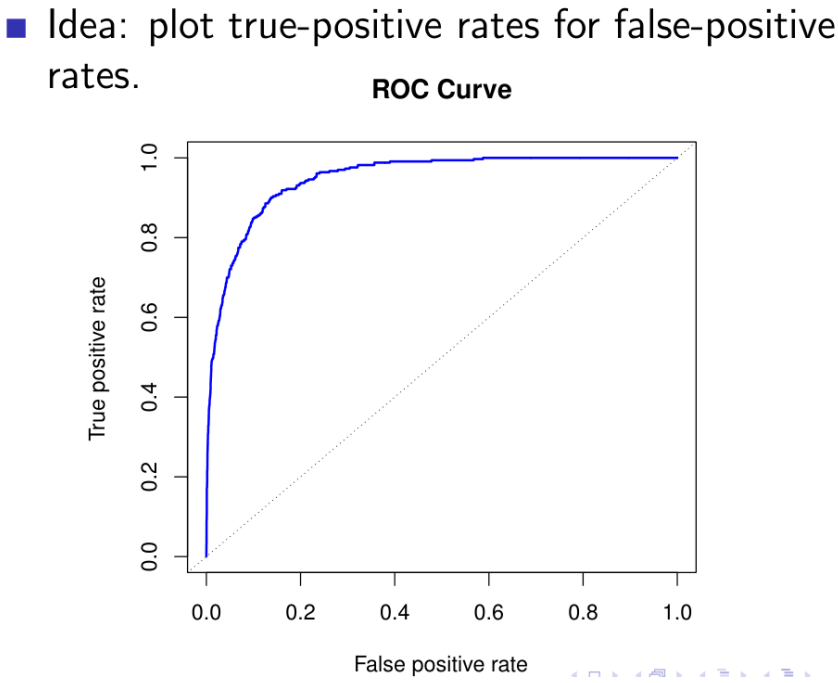

Figura 5: Curva ROC.

Podemos selecionar o melhor α como sendo por exemplo o mais próximo de $(0, 1)$. Para um dado classificador, podemos computar a área abaixo da curva ROC, *area under the curve* AUC. O melhor AUC é 1. O AUC 1/2 é muito ruim. Na realidade, o AUC é uma estimativa de $P(h(x') > h(x'') \mid Y' = 1, Y'' = 0)$.

2.3 Estatística frequentista

A estimação de máxima verossimilhança é

$$\hat{\theta} = \arg\max p(x|\theta),$$

onde $L(\theta) = p(x|\theta)$ é a probabilidade de fazer as observações x dado o parâmetro θ. A partir das observações, nós obtemos uma estimativa para θ tentando maximizar a probabilidade de obtermos as observações que de fato observamos.

Para variáveis independentes (o caso em que geral trataremos):

$$L(\theta) = p(x_1, \ldots, x_n|\theta) = \prod_{i=1}^{N} p(x_i|\theta).$$

Para variáveis gaussianas $X_1, \ldots, X_n$ i.i.d. que tiveram observações $x_1, \ldots, x_n$, o princípio de máxima verossimilhança nos dá $\hat{\theta} = \frac{1}{N}\sum_i x_i$.

Um intervalo $(1-\alpha)$ de confiança para θ é o intervalo $[a, b]$ tal que

$$P(\theta \in [a, b]) \geq 1 - \alpha,$$

para todo θ.

Para pegarmos o z em função da probabilidade dividido por 2, basta olhar a tabela da Figura 6. Note que z é apenas uma parametrização, onde a integral de μ até $\mu + z\sigma$ da gaussiana dá o respectivo valor da tabela. Como um exemplo, caso queiramos o $z = 1.96$, nós olhamos para a linha 1.9 e coluna 0.06 e encontramos o valor 0.4750 que é o meio da probabilidade, ou seja $0.4750 \times 2 = 95\%$ do intervalo de confiança. Para uma distribuição normal, este é $\left[\mu - \frac{z\sigma}{\sqrt{N}}, \mu + \frac{z\sigma}{\sqrt{N}}\right]$.

Distribuição Normal P(0≤Z<z0)										
z0	**0,00**	**0,01**	**0,02**	**0,03**	**0,04**	**0,05**	**0,06**	**0,07**	**0,08**	**0,09**
0,0	0,0000	0,0040	0,0080	0,0120	0,0160	0,0199	0,0239	0,0279	0,0319	0,0359
0,1	0,0398	0,0438	0,0478	0,0517	0,0557	0,0596	0,0636	0,0675	0,0714	0,0753
0,2	0,0793	0,0832	0,0871	0,0910	0,0948	0,0987	0,1026	0,1064	0,1103	0,1141
0,3	0,1179	0,1217	0,1255	0,1293	0,1331	0,1368	0,1406	0,1443	0,1480	0,1517
0,4	0,1554	0,1591	0,1628	0,1664	0,1700	0,1736	0,1772	0,1808	0,1844	0,1879
0,5	0,1915	0,1950	0,1985	0,2019	0,2054	0,2088	0,2123	0,2157	0,2190	0,2224
0,6	0,2257	0,2291	0,2324	0,2357	0,2389	0,2422	0,2454	0,2486	0,2517	0,2549
0,7	0,2580	0,2611	0,2642	0,2673	0,2704	0,2734	0,2764	0,2794	0,2823	0,2852
0,8	0,2881	0,2910	0,2939	0,2967	0,2995	0,3023	0,3051	0,3078	0,3106	0,3133
0,9	0,3159	0,3186	0,3212	0,3238	0,3264	0,3289	0,3315	0,3340	0,3365	0,3389
1,0	0,3413	0,3438	0,3461	0,3485	0,3508	0,3531	0,3554	0,3577	0,3599	0,3621
1,1	0,3643	0,3665	0,3686	0,3708	0,3729	0,3749	0,3770	0,3790	0,3810	0,3830
1,2	0,3849	0,3869	0,3888	0,3907	0,3925	0,3944	0,3962	0,3980	0,3997	0,4015
1,3	0,4032	0,4049	0,4066	0,4082	0,4099	0,4115	0,4131	0,4147	0,4162	0,4177
1,4	0,4192	0,4207	0,4222	0,4236	0,4251	0,4265	0,4279	0,4292	0,4306	0,4319
1,5	0,4332	0,4345	0,4357	0,4370	0,4382	0,4394	0,4406	0,4418	0,4429	0,4441
1,6	0,4452	0,4463	0,4474	0,4484	0,4495	0,4505	0,4515	0,4525	0,4535	0,4545
1,7	0,4554	0,4564	0,4573	0,4582	0,4591	0,4599	0,4608	0,4616	0,4625	0,4633
1,8	0,4641	0,4649	0,4656	0,4664	0,4671	0,4678	0,4686	0,4693	0,4699	0,4706
1,9	0,4713	0,4719	0,4726	0,4732	0,4738	0,4744	0,4750	0,4756	0,4761	0,4767
2,0	0,4772	0,4778	0,4783	0,4788	0,4793	0,4798	0,4803	0,4808	0,4812	0,4817
2,1	0,4821	0,4826	0,4830	0,4834	0,4838	0,4842	0,4846	0,4850	0,4854	0,4857
2,2	0,4861	0,4864	0,4868	0,4871	0,4875	0,4878	0,4881	0,4884	0,4887	0,4890
2,3	0,4893	0,4896	0,4898	0,4901	0,4904	0,4906	0,4909	0,4911	0,4913	0,4916
2,4	0,4918	0,4920	0,4922	0,4925	0,4927	0,4929	0,4931	0,4932	0,4934	0,4936
2,5	0,4938	0,4940	0,4941	0,4943	0,4945	0,4946	0,4948	0,4949	0,4951	0,4952
2,6	0,4953	0,4955	0,4956	0,4957	0,4959	0,4960	0,4961	0,4962	0,4963	0,4964
2,7	0,4965	0,4966	0,4967	0,4968	0,4969	0,4970	0,4971	0,4972	0,4973	0,4974
2,8	0,4974	0,4975	0,4976	0,4977	0,4977	0,4978	0,4979	0,4979	0,4980	0,4981
2,9	0,4981	0,4982	0,4982	0,4983	0,4984	0,4984	0,4985	0,4985	0,4986	0,4986
3,0	0,4987	0,4987	0,4987	0,4988	0,4988	0,4989	0,4989	0,4989	0,4990	0,4990

Figura 6: Tabela de intervalo de confiança.

Para o erro empírico $\hat{e}_r$ a partir de M observações, ainda sim olhamos a tabela da Figura 6 para descobrimos a relação entre z e o valor do intervalo de confiança. Então

usamos a fórmula

$$\left[\hat{e}_r - z\sqrt{\frac{\hat{e}_r(1-\hat{e}_r)}{M}}, \hat{e}_r + z\sqrt{\frac{\hat{e}_r(1-\hat{e}_r)}{M}}\right],$$

que indica a taxa de erro de probabilidade $1-\alpha$ que produz o z de acordo com a tabela.

2.3.1 Teste de hipótese

Começamos com uma hipótese nula $H_0 : \theta \in \Theta_0$ e uma hipótese alternativa $H_1 : \theta \in \Theta_1$. Então coletamos dados e checamos se a hipótese nula é rejeitada: rejeitamos-a quando os dados caem dentro de uma região de rejeição R. Existem dois tipos de erro nesse processo:

- Erro tipo I: quando se rejeita H_0 e ele é verdadeiro.
- Erro tipo II: não se rejeita H_0 quando ele é falso.

A ideia é escolher uma região de rejeição R que controle o erro tipo I. O nível de R é α quando a probabilidade de erro tipo I é menor ou igual que α, dado que H_0 é verdadeiro.

Exemplo: $X_i \sim N(\theta, \sigma)$.

Teste:

$$\begin{cases} H_0 : \theta \leq 0, \\ H_1 : \theta > 0. \end{cases}$$

Região de rejeição

$$R = \left\{ x_1, \ldots, x_n : \frac{1}{N}\sum_i x_i > c \right\}.$$

Escolhemos c de maneira que $P(X \in R \mid \theta) \leq \alpha$ para todo $\theta \in H_0$. Um teste com o menor erro do tipo II para cada α é o *teste mais forte*.

O conceito de $p-$*valor* é o seguinte:

- Faça um teste: rejeite H_0 se, e somente se, $T(x_1, \ldots, x_N) \geq c$.
- O $p-$valor é
  - o menor nível α em que podemos rejeitar H_0.
  - ou a máxima probabilidade de que $T(X_1, \ldots, X_N)$ é maior que $T(x_1, \ldots, x_N)$ quando H_0 é verdadeiro.
- um $p-$valor muito pequeno é uma evidência forte contra H_0.
- tipicamente < 0.01 é muito forte contra H_0, já com > 0.1 tem-se pouca evidência contra H_0.

Suponha que tenhamos dois classificadores e um dataset de teste. Queremos testar se eles têm taxas de erro iguais. Nós fazemos validação cruzada $k-$fold, e então aplicamos ambos os clasificados em cada fold. Isso nos dá k pares $(a_1, b_1), \ldots, (a_k, b_k)$. Então assumimos que a distribuição de acurácias é Gaussiana com variâncias iguais, mas desconhecidas. A hipótese nula é que não tem diferença.

Wald test com α: rejeitar H_0 se $|W| > z$, onde $P(-z \leq Z \leq z) = 1 - \alpha$ e

$$W = \frac{\mu}{s},$$

onde $\mu = \sum_{i=1}^{N}(a_i - b_i)/N$ e

$$s = \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^{N} \left(a_i - b_i - \frac{1}{N} \sum_{j=1}^{N} (a_j - b_j) \right)^2}.$$

Como W é assintoticamente gaussiano, temos o intervalo de confiança para a diferença entre os dois classificadores $[\mu - zs, \mu + zs]$, onde z é relacionado a $1 - \alpha$ pela Figura 6.

3 Regression

Lembre que temos o modelo $Y = f(X) + \epsilon$, onde ϵ é independente de f e tem média zero. Regressão linear adota $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$.

O método de mínimos quadrados é:

- Suponha que tenhamos as observações $x_{1j}, x_{2j}, \ldots, x_{nj}, y_j$, para $j \in \{1, \ldots, N\}$.
- Consideremos o *residual sum of squares* que tentaremos minimizar pelos parâmetros $\beta_0, \beta_1, \ldots, \beta_n$.

$$\text{RSS} = \sum_{j=1}^{N} (y_j - f(x_j))^2.$$

3.1 Regressão Linear Simples

Nós temos $Y = \beta_0 + \beta_1 X_1$. Para minimizar RSS:

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}_1,$$

$$\hat{\beta}_1 = \frac{\sum_j (x_{1j} - \overline{x}_1)(y_j - \overline{y})}{\sum_j (x_{1j} - \overline{x}_1)^2},$$

onde $\overline{x}_1 = \frac{1}{N} \sum_{j=1}^{N} x_{1j}$ e $\overline{y} = \frac{1}{N} \sum_{j=1}^{N} y_j$.

A estimativa $\hat{\sigma}^2$ *unbiased* para σ^2 deve levar em conta os dois graus de liberdade. Logo

$$\hat{\sigma}^2 = \frac{1}{N-2} \sum_{j=1}^{N} (y_j - \hat{y}_j)^2.$$

Existe um teste padrão para $H_0 : \beta_1 = 0$ versus $H_1 : \beta_1 \neq 0$. Rejeite H_0 quando

$$\left| \frac{\hat{\beta}_1}{\sqrt{\hat{\sigma}^2 \sum_{j=1}^{N} (x_{1j} - \overline{x}_1)^2}} \right| > z.$$

O $p-$valor para esse caso funciona assim. Faça um teste da forma $T > c$ para T que dependa dos dados, e algum c. Então um $p-$valor é a probabilidade que T seja maior que o T observado, para H_0 verdadeiro (ou seja, $\beta_1 = 0$). Lembrando que $p-$valor pequeno é uma evidência contra H_0.

Uma medida comum do fit é o R^2. No caso geral $X_1, X_2, \ldots, X_n$:

$$R^2 = 1 - \frac{\text{RSS}}{\sum_{j=1}^{N}(y_j - \overline{y})^2},$$

que é a proporção da variabilidade do Y que pode ser explicada por $X_1, \ldots, X_n$. R^2 está entre 0 e 1. Quando maior melhor.

No caso de uma variável X_1 somente, temos a expressão

$$R^2 = \left(\frac{\sum_{j=1}^{N}(x_{1j} - \overline{x}_1)(y_j - \overline{y})}{\sqrt{\sum_{j=1}^{N}(x_{1j} - \overline{x}_1)^2}\sqrt{\sum_{j=1}^{N}(y_j - \overline{y})^2}} \right)^2 = \left(\frac{E[(X_1 - E(X_1))(Y - E(Y))]}{\sigma_{X_1}\,\sigma_Y} \right)^2 =$$

$$= \frac{\text{cov}(X_1, Y)^2}{\sigma_{X_1}^2 \sigma_Y^2} = \rho_{X_1,Y}^2.$$

Note que isso é a correlação entre X_1 e Y.

3.2 Caso mais geral

Para muitas variáveis $Y = \beta_1 X_1 + \cdots + \beta_n X_n + Z$ (note que $\beta_0 = 0$), com $E[Z] = 0$, o RSS é $(C - AB)^T(C - AB)$, onde

$$A = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1N} & x_{2N} & \cdots & x_{nN} \end{pmatrix}, \quad B = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix}, \quad C = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix},$$

e os β_i que minimizam o RSS são

$$\hat{B} = (A^T A)^{-1} A^T C.$$

Se Z tem variância σ^2, então $\hat{B}$ tem variância $\sigma^2(A^T A)^{-1}$ e é aproximadamente gaussiano para grande N.

Novamente, podemos testar a variabilidade do Y com $X_1, X_2, \ldots, X_n$ através do

$$R^2 = 1 - \frac{\text{RSS}}{\sum_{j=1}^{N}(y_j - \overline{y})^2}.$$

3.3 Covariate selection

Geralmente, devemos descartar algumas covariáveis. Assim, devemos dar um *score* para cada conjunto de variáveis, e escolher o melhor. Um score popular é o AIC (Akaike Information Criterion):

$$L_S - |S|, \text{ (goodness of fit} - \text{model complexity)},$$

onde S é o conjunto de covariáveis no modelo scored e L_S é a log-likelihood do modelo com covariáveis em S, avaliada na estimação de máxima verossimilhança.

Outro popular é o BIC (Bayesian Information Criterion):

$$L_S - \frac{|S|}{2} \log N.$$

Para N grande, a probabilidade posterior do scored model é proporcional a e^{BIC}, quando todos os modelos têm probabilidade idêntica.

Outro score é o *Mallow's* C_p:

$$2|S|\hat{\sigma}^2 + \sum_{j=1}^{N} (\hat{y}_j^S - y_j)^2,$$

onde $\hat{\sigma}^2$ é a estimativa da unbiased variância com todas as covariáveis, enquanto $\hat{y}_j^S$ são produzidos com covariáveis somente em S. Esse score estima o erro de treino que é esperado de um modelo com covariáveis em S.

Structure search:

- Forward stepwise regression: começar sem covariáveis, adicionar uma que leva ao melhor score, depois adicionar outra que leva ao melhor score, etc.
- Backward stepwise regression: começar com todas as covariáveis, retirar aquela que leva ao melhor score, etc.

Quando se tem features qualitativas:

- Fazer algum encoding (transformar em números, ou outros data structures).
- Se os valores forem ordenados: transformá-los em números.
- Se não for apropriado ordená-los: *one-hot* encoding. Ou seja, para cada valor único numa coluna qualitativa, uma nova coluna é criada. Então essas novas variáveis são preenchidas com zeros e uns (1 significando `True` e 0 `False`).

Nonlinear terms in regression: colocar termos que não são lineares, LOL.
Polynomial regression: quando as funções são polinômios das covariáveis.

3.4 Regularização

A ideia é penalizar o "tamanho" dos parâmetros, para reduzir a variância (mas com um aumento no bias). Isso é útil para reduzir overfitting. As duas principais estratégias são *ridge regression* e *lasso*.

No Ridge a gente só adiciona um novo termo $\lambda \sum_{i=1}^{N} \beta_i^2$. Agora minimazeremos

$$\sum_{j=1}^{N} \left(y_j - \sum_{i=1}^{n} \beta_i x_{ij} \right)^2 + \lambda \sum_{i=1}^{n} \beta_i^2.$$

Quanto maior o parâmetro $\lambda \geq 0$ menor os valores de $\hat{\beta}_i$. Para tunar λ geralmente se faz cross-validation.

A solução é fácil (mas biased):

$$\hat{B} = (A^T A + \lambda I)^{-1} A^T C.$$

Em regressão linear, multiplicar as covariáveis leva à estimativas multiplicadas. Assim, uma suposição usual é que os dados estejam normalizados (média 0 e variância 1):

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_{ij} - \overline{x}_i)^2}},$$

e que Y é centrado (a média é subtraída).

Na regressão Lasso nós minimizamos

$$\sum_{j=1}^{N}\left(y_j - \sum_{i=1}^{n}\beta_i x_{ij}\right)^2 + \lambda\sum_{i=1}^{n}|\beta_i|,$$

e novamente tunamos λ com cross-validation, normalizamos X_i e centralizamos Y.

Amazing fact: se λ é grande o bastante, muitos dos β_i são setados à zero. Logo, feature selection é feita automaticamente.

Com o lasso, não existe solução analítica fechada. Mas dá para fazer a minimazação com otimização convexa.

Tanto Ridge quanto Lasso minimizam RSS, mas

- Ridge: sujeito a $\sum_i \beta_i^2 \leq \tilde{\lambda}$.
- Lasso: sujeito a $\sum_i |\beta_i| \leq \tilde{\lambda}$.

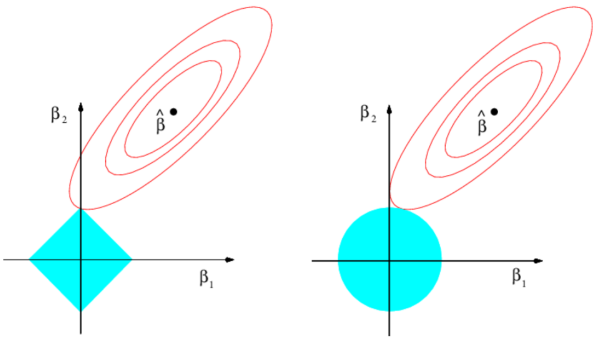

Figura 7: Ridge vs Lasso.

4 Logistic regression

Suponha que Y seja binário (valores 0 ou 1, problema de classificação). Nós assumimos que

$$P(Y = 1 \mid X_1, \ldots, X_n) = \frac{e^W}{1 + e^W},$$

onde

$$W = \sum_{i=1}^{n}\beta_i X_i.$$

Equivalentemente

$$\text{logit}(P(Y = 1 \mid X_1, \ldots, X_n)) = \sum_{i=1}^{n} \beta_i X_i,$$

onde $\text{logit}(p) = \log(p/(1-p))$.

A fronteira

$$\frac{e^W}{1+e^W} = 1/2$$

é dada pela equação linear $W = \sum_{i=1}^{n} \beta_i X_i = 0$.

A likelihood function dadas observações x_{ij}, y_j é

$$L(\beta_1, \ldots, \beta_n) = \prod_{j=1}^{N} \left(\frac{e^{\sum_{i=1}^{N} \beta_i x_{ij}}}{1 + e^{\sum_{i=1}^{N} \beta_i x_{ij}}} \right)^{y_j} \left(1 - \frac{e^{\sum_{i=1}^{N} \beta_i x_{ij}}}{1 + e^{\sum_{i=1}^{N} \beta_i x_{ij}}} \right)^{1-y_j}.$$

Problema: não existe solução fechada para o máximo da função de likelihood acima. Um algoritmo popular para resolver a maximização acima é o IRLS (Iterative Reweighted Least Squares Algorithm).

4.1 IRLS

1. Comece com um chute $\hat{\beta}_i^0$.
2. Repita até a convergência:
   (a) $p_j \leftarrow \frac{e^{\sum_{i=1}^{N} \beta_i x_{ij}}}{1+e^{\sum_{i=1}^{N} \beta_i x_{ij}}}$.
   (b) $q_j \leftarrow \text{logit}(p_j) + \frac{y_j - p_j}{p_j(1-p_j)}$.
   (c) $F \leftarrow \text{diag}\Big(p_1(1-p_1), p_2(1-p_2), \ldots, p_N(1-p_N)\Big)$.
   (d) Coloque

$$\begin{pmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_n \end{pmatrix} \leftarrow (A^T F A)^{-1} A^T F \begin{pmatrix} q_1 \\ \vdots \\ q_N \end{pmatrix}.$$

4.2 LDA

Suponha que tenhamos uma classe binária e uma feature contínua, e

- X dado $Y = 0$ é gaussiana $\mathcal{N}(\mu_0, \sigma^2)$.
- X dado $Y = 1$ é gaussiana $\mathcal{N}(\mu_1, \sigma^2)$.

Para obter $P(Y = 1 \mid x)$, utilizar a regra de Bayes

$$P(Y = 1 \mid x) = \frac{p(x \mid Y = 1)P(Y = 1)}{\sum_k p(x \mid Y = k)P(Y = k)}.$$

Então o classificador de Bayes é

$$\begin{cases} 1 \text{ se } \log\left(\frac{P(Y=1)}{P(Y=0)}\right) - r_1^2 + r_0^2 > 0 \\ 0 \text{ caso contrário,} \end{cases}$$

$$\iff \begin{cases} 1 \text{ se } 2x(\mu_1 - \mu_0) > \mu_1^2 - \mu_0^2 - 2\sigma^2 \log\left(\frac{P(Y=1)}{P(Y=0)}\right) \\ 0 \text{ caso contrário,} \end{cases}$$

onde $r_i^2 = (x - \mu_i)^2/2\sigma^2$. Note que a expressão acima é linear.

4.3 QDA

Mesmas hipótese que no LDA, mas agora X dado $Y = 0$ ou $Y = 1$ tem variâncias diferentes σ_0^2 e σ_1^2. O classificador Bayes é

$$\begin{cases} 1 \text{ se } r_1^2 - r_0^2 < \log\left(\frac{\sigma_0 P(Y=1)}{\sigma_1 P(Y=0)}\right) \\ 0 \text{ caso contrário.} \end{cases}$$

Então o QDA é: use o classificador plug-in de máxima verossimilhança.

4.4 Comparação

- Para o LDA, com dois labels, a fronteira de classificação é uma função linear dos features.
- Assim, o LDA tem o mesmo comportamento que a regressão logística.
  - Mas o LDA é construído estimando as probabilidades de Y e das features, enquanto que a regressão logística foca em estimar os coeficientes de fronteira.
  - Regressão logística é geralmente melhor que LDA.

Observações:

- LDA, Naive Bayes etc, são *generativos*: eles tentam modelar $P(X, Y)$.
- Regressão logística é *discriminativo*: ela teta modelar $P(Y \mid X)$.
- A primeira é mais "perto da verdade" ??, mas a última é geralmente mais exata.

5 SVM

Consideremos um problema classificação binário com $Y \in \{-1, 1\}$.
Suponha que as classes (Y) são separáveis linearmente por um hiperplano

$$H(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n = 0,$$

ou seja, temos esse tipo de situação

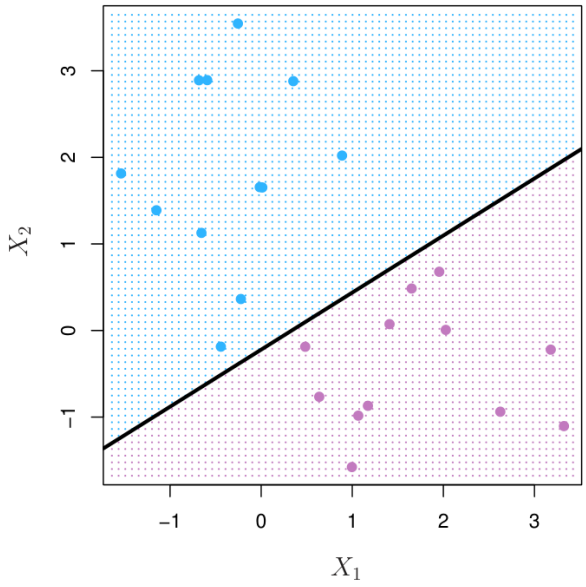

Figura 8: Separação linear das classes.

Quando existirem vários hiperplanos possíveis que separam os dados, pegue o hiperplano com margem máxima. A margem é a distância do hiperplano para o ponto de treino mais próximo.

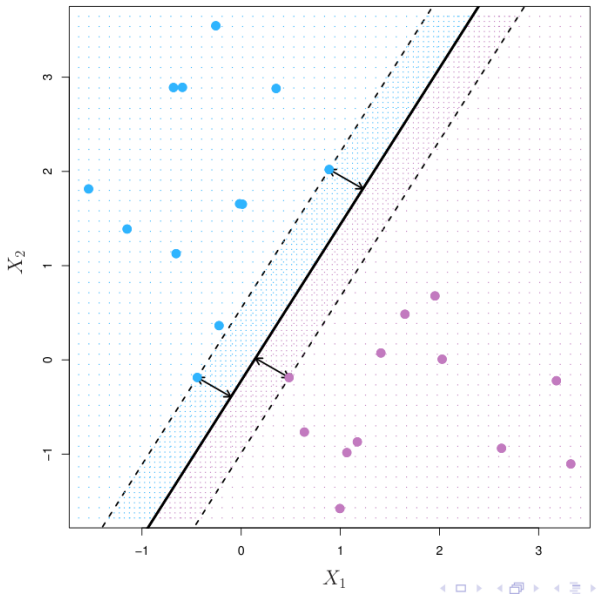

Figura 9: Escolha do plano com maior margem.

Para o hiperplano com margem maximal, os vetores suportes são os pontos que estão mais próximos dele. Para tal hiperplano, a distância da observação i até o hiperplano é dada por

$$\frac{y_j(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_n x_{nj})}{\sqrt{\beta_1^2 + \cdots + \beta_n^2}}.$$

Então tente achar $\beta_0, \beta_1, \ldots, \beta_n$ que maximize M tal que

$$\sum_{i=1}^{n} \beta_i^2 = 1 \quad \text{e} \quad y_j(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_n x_{nj}) \geq M, \forall\, j.$$

Esse tipo de problemas de otimização pode ser resolvido rapidamente utilizando programação quadrática. O mínimo é garantido de ser achado.

Na prática, claramente os problemas são não-separáveis. Se um problema é não-separável, então não há solução com $M > 0$. O problema então deve ser relaxado. Considere uma *softmargin*: deixemos algumas observação violar a margem. Daí a situação fica com margens soft pontilhadas:

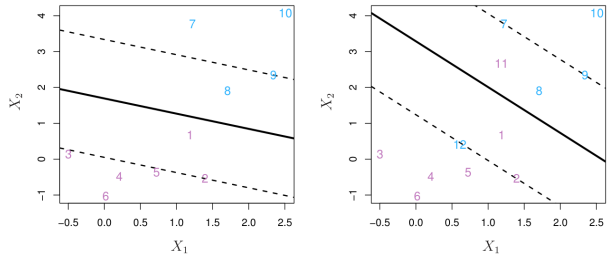

Figura 10: Margem soft em pontilhado.

O problema de otimização *soft* é então achar $\beta_0, \beta_1, \ldots, \beta_n$ e $\epsilon_1, \ldots, \epsilon_N$ que maximizam M sujeitos a

$$\sum_{i=1}^{n} \beta_i^2 = 1 \quad \text{e} \quad y_j(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_n x_{nj}) \geq M(1 - \epsilon_j),$$

para cada j e

$$\epsilon_j \geq 0, \quad \sum_{j=1}^{N} \epsilon_j \leq C.$$

Alguns pontos:

- O problema soft pode ser resolvido por programação quadrática.
- Importante: apenas observações dentro da margem afetam o hiperplano.
  - Apenas as fronteiras importam.
  - SVMs estão mais próximas de classificadores discriminativos do que generativos.
- O hiperparâmetro C é geralmente tunado por cross-validation.
  - Se C é zero, não relaxação; Quanto maior C, mais pontos de treino podem estar dentro da margem.

6 Nonlinear boundaries

Podemos capturar fronteiras não lineares engrandecendo as features, por exemplo

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_n, X_n^2,$$

e talvez outras funções de features.

A estrutura das SVMs permitem fazer isso eficientemente para espaços muito engrandecidos, usando os *kernels*.

Acontece que SVMs podem ser aprendidas apenas tratando dos produtos internos $x' \cdot x''$, onde x' e x'' são duas observações. Suponha que tenhamos funções $\phi(X) = [\phi_1(X), \ldots, \phi_m(X)]$, então precisaríamos das quantidades $\phi(x') \cdot \phi(x'')$. Ao invés disso, podemos apenas utilizar a função $K(X', X'')$ toda vez que um produto escalar é necessário. Tal função é chamada *kernel*.

Popular kernels:

- Polinomial: $K(X', X'') = (1 + X' \cdot X'')^d$.
- Base radial: $K(X', X'') = \exp(-\gamma \|X' \cdot X''\|)$.
- Neural: $K(X', X'') = \tanh(\gamma_1(X' \cdot X'') + \gamma_2)$.

Os hiperparâmetros dos kernels são, como sempre, tunados por cross-validation.

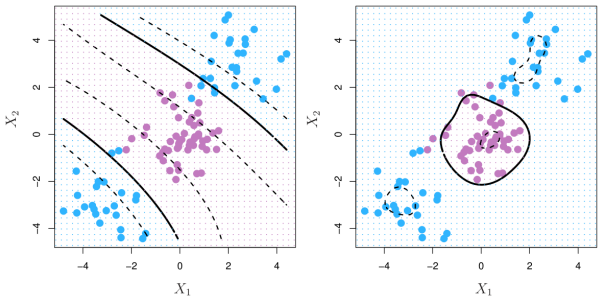

Figura 11: SVM quando kernels são aplicados.