5
Algoritmos baseados em funções de Núcleo

Os algoritmos para regressão PLS apresentados até agora modelam uma variável a ser predita Y de forma linear via regressão por mínimos quadrados. A característica do PLS de realizar a regressão de forma parcial em cada fator de X fornece melhor resultado do que a regressão convencional OLSR[1] [30] que emprega todas as variáveis simultaneamente.

Para certos conjuntos de dados, o uso de um modelo não-linear fornece uma melhor qualidade preditiva. Existem várias versões não lineares do algoritmo para regressão PLS, por exemplo, em [54] é proposta uma modelagem polinomial quadrática entre os escores t e a variável dependente Y. Outro exemplo é encontrado em [42, 34], onde a não linearidade é introduzida projetando as amostras em curvas encontradas via redes neurais artificiais.

Neste capítulo, são descritas variantes não lineares da regressão PLS baseadas em funções de núcleo [47]. A redefinição do produto interno com uma função de núcleo acarreta num mapeamento implícito das amostras num espaço de maior dimensão. Esta abordagem evita certas dificuldades de outros métodos não lineares [2, 3], pois a modelagem continua sendo linear, só que aplicada à nova representação dos dados. Com isto, o algoritmo continua com a mesma complexidade operacional do algoritmo original PLS, e a não linearidade é ajustada com a escolha da função de núcleo apropriada para a aplicação.

Na seção seguinte, é fornecida uma explicação de funções de núcleo, suficiente para o entendimento dos algoritmos descritos nas próximas seções.

5.1
Funções de Núcleo

O objetivo ao se representar um algoritmo linear via funções de núcleo é fornecer um mapeamento aos dados de entrada, de forma que a tarefa de

[1] Ordinary Least Squares Regression

aprendizagem seja facilitada. Considerando que os dados de entrada $\mathsf{x}_i \in \mathbb{R}^m$ $1 \leq i \leq n$, pertencem ao espaço vetorial $\mathcal{X}$, e estabelecendo $\mathcal{F}$ como o espaço obtido com os dados na sua nova representação, definimos Φ como sendo o mapeamento

$$\begin{aligned} \Phi: \quad & \mathcal{X} \mapsto \mathcal{F} \\ & \mathsf{x} \ \mapsto \Phi(\mathsf{x}) \end{aligned}$$

Procuramos com este mapeamento uma nova representação dos dados x_i que forneça as características necessárias para uma melhor aprendizagem. Para o caso de uma regressão linear de v em u por exemplo, a figura 5.1 ilustra um resultado desejado.

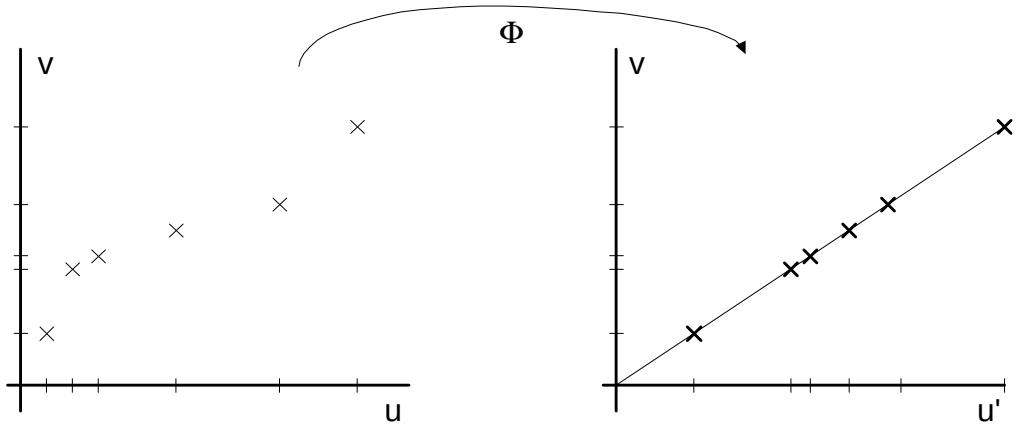

Figura 5.1: Exemplo de mapeamento desejado.

A dimensão M para o espaço $\mathcal{F} = \{\Phi(\mathsf{x}) \mid \mathsf{x} \in \mathcal{X}\}$ contendo as características[2] é conseqüência do mapeamento escolhido. Se for usada uma projeção num conjunto reduzido de componentes principais, como é o caso com a a transformada de Hotelling para compressão de imagens [18], a dimensão do espaço é reduzida, sendo $M \ll m$. Porém para o algoritmo PLS, procuramos justamente um mapeamento que acrescente variáveis capazes de descrever a não linearidade existente entre as variáveis dependentes e independentes, resultando num aumento da dimensão, $M \gg m$.

Um problema ao se acrescentar variáveis está no aumento da complexidade computacional, mas ao se reformular o algoritmo via funções de núcleo, esta dificuldade é contornada.

[2]Feature Space

5.1.1 Mapeamento

A adição de novas variáveis se torna computacionalmente viável, pois ao invés de representar explicitamente os dados no espaço das características $\mathcal{F}$, o produto interno neste espaço é expresso por funções de núcleo no espaço de entrada $\mathcal{X}$. Ou seja, uma função de núcleo K é definida por

$$K(\mathsf{u}, \mathsf{v}) = \langle \Phi(\mathsf{u}) \cdot \Phi(\mathsf{v}) \rangle \tag{5-1}$$

onde $\mathsf{u}, \mathsf{v} \in \mathcal{X}$ e $\langle \Phi(\mathsf{u}) \cdot \Phi(\mathsf{v}) \rangle$ representa o produto interno entre a representação das amostras. Desta maneira, se o algoritmo linear for reformulado de tal forma que as amostras não sejam expressas explicitamente mas somente o produto interno entre elas, o uso de uma função de núcleo (eq. 5-1) apropriada fornece um mapeamento implícito. Nos algoritmos, isto é realizado com o uso da matriz núcleo[3] K onde

$$\mathsf{K}_{ij} = K(\mathsf{x}_i, \mathsf{x}_j) \tag{5-2}$$

com $\mathsf{x}_i, \mathsf{x}_j \in \mathcal{X}$ e $1 \leq i, j \leq n$, n sendo o número de amostras.

Por exemplo, a função

$$K(\mathsf{u}, \mathsf{v}) = (\mathsf{u}^\top \mathsf{v} + 1)^3 \tag{5-3}$$

com $\mathsf{u}, \mathsf{v} \in \mathbb{R}^2$, $\mathsf{u}^\top = (x_1, y_1)$ e $\mathsf{v}^\top = (x_2, y_2)$, fornece

$$\begin{aligned} K(\mathsf{u}, \mathsf{v}) &= (x_1 x_2 + y_1 y_2 + 1)^3 \\ &= 1 + 3x_1 x_2 + 3y_1 y_2 + 6x_1 x_2 y_1 y_2 \\ &\quad + 3x_1^2 x_2^2 + 3y_1^2 y_2^2 + 3x_1 x_2 y_1^2 y_2^2 + 3x_1^2 x_2^2 y_1 y_2 \\ &\quad + x_1^3 x_2^3 + y_1^3 y_2^3 \end{aligned}$$

[3] Também chamada de matriz Gram.

o que pode ser visto como o produto interno das amostras, $\Phi(\mathsf{u})^\top\Phi(\mathsf{v})$, na seguinte representação

$$(1, \sqrt{3}x_1, \sqrt{3}y_1, \sqrt{6}x_1y_1, \sqrt{3}x_1^2, \sqrt{3}y_1^2, \sqrt{3}x_1y_1^2, \sqrt{3}x_1^2y_1, x_1^3, y_1^3) \begin{pmatrix} 1 \\ \sqrt{3}x_2 \\ \sqrt{3}y_2 \\ \sqrt{6}x_2y_2 \\ \sqrt{3}x_2^2 \\ \sqrt{3}y_2^2 \\ \sqrt{3}x_2y_2^2 \\ \sqrt{3}x_2^2y_2 \\ x_2^3 \\ y_2^3 \end{pmatrix}$$

Com isto, os dados que originalmente estavam em $\mathbb{R}^2$ foram mapeados para $\mathcal{F} \subseteq \mathbb{R}^{10}$, sendo introduzida a não linearidade desejada com monômios de até grau 3. O termo constante na equação (5-3) faz com que os termos originais do produto interno sejam mantidos. É importante notar que o mapeamento mostrado não é o único possível para a função de núcleo da equação (5-3), qualquer representação que forneça o mesmo produto interno é válida.

5.1.2 Funções usadas

A escolha da função de núcleo para a não linearização do algoritmo de regressão PLS requer que o seguinte problema seja resolvido: como definir uma função garantindo que seja uma função de núcleo associada a um espaço de características $\mathcal{F}$.

A garantia de que uma dada função é válida é conseguida com o teorema de Mercer [6, 4]. Nesta tese, são usadas duas funções de núcleo clássicas: polinomial e gaussiana, sendo a aplicação do teorema facilmente encontrada na literatura [4].

Núcleo polinomial

A função de núcleo polinomial é definida por

$$K(\mathsf{u}, \mathsf{v}) = (\mathsf{u}^\top\mathsf{v} + c)^d, \quad \mathsf{u}, \mathsf{v} \in \mathbb{R}^m \tag{5-4}$$

gerando todos os monômios de grau até d. Desta forma, o uso de um núcleo de grau 2 para uma regressão linear tende a valorizar a relação quadrática existente entre as variáveis originais e a dependente. Como provado em [6, 4], o emprego desta função como produto interno leva à representação de amostras contidas num espaço de dimensão P_{dim} onde

$$P_{dim} = \begin{pmatrix} m+d \\ d \end{pmatrix} \tag{5-5}$$

Núcleo gaussiano

Quanto à função gaussiana, sua definição é dada por

$$K(\mathsf{u},\mathsf{v}) = e^{-\frac{\|\mathsf{u}-\mathsf{v}\|^2}{\sigma^2}}, \quad \mathsf{u},\mathsf{v} \in \mathbb{R}^m$$

Uma forma de interpretá-la é reparar que para dado σ, duas amostras são ortogonais caso estejam distantes, ou seja, $\|\mathsf{u}-\mathsf{v}\|$ seja relativamente grande quando comparado com σ. Como mostrado em [4], dependendo do parâmetro, este tipo de núcleo leva a um mapeamento implícito num espaço de alta dimensão, podendo até ser infinita. Seu uso em classificadores SVM[4] [6] simula o funcionamento de uma rede RBF[5] [50].

Propriedades

Além dos dois núcleos apresentados, foram usadas as seguintes propriedades [6], cada uma delas podendo ser usada para a construção de uma nova função de núcleo K:

1. $K(\mathsf{u},\mathsf{v}) = K_1(\mathsf{u},\mathsf{v}) + K_2(\mathsf{u},\mathsf{v})$
2. $K(\mathsf{u},\mathsf{v}) = aK_1(\mathsf{u},\mathsf{v})$
3. $K(\mathsf{u},\mathsf{v}) = K_1(\mathsf{u},\mathsf{v})K_2(\mathsf{u},\mathsf{v})$

onde K_1 e K_2 são duas funções de núcleo dadas com $\mathsf{u},\mathsf{v} \in \mathcal{X}$, $\mathcal{X} \subseteq \mathbb{R}^m$ e $a \in \mathbb{R}^+$.

[4] Support Vector Machines.
[5] Radial Basis Function.

5.1.3 Aplicação

O uso de funções de núcleo com algoritmos lineares é computacionalmente viável, porque os dados não são representados explicitamente no espaço $\mathcal{F}$. Sua representação é implícita via redefinição do produto interno em $\mathcal{X}$. De fato, o uso de um núcleo polinomial de grau 4 numa aplicação comum do algoritmo PLS, onde as amostras possuem 300 variáveis, resulta num mapeamento (eq. 5-5) para um espaço de 348.881.876 dimensões, tornando a representação explícita inviável.

A dificuldade no uso de funções de núcleo está na reformulação do algoritmo, expressando apenas o produto interno entre as amostras através da matriz núcleo[6] K, sem usar explicitamente as variáveis adicionais. A presença de $\mathcal{F}$ fornece uma garantia teórica para os resultados.

O poder da técnica reside no fato de o algoritmo de aprendizagem permanecer linear se for usada como mapeamento a identidade, com $\mathsf{K}_{ij} = \mathsf{x}_i^\top \mathsf{x}_j$. Neste caso, o algoritmo possui o mesmo comportamento do original linear. Com isto, a introdução da não-linearidade se faz com a escolha da função de núcleo apropriada para a aplicação em questão.

5.1.4 KBPLS

Como exemplo de aplicação, temos o algoritmo para regressão PLS baseado em funções de núcleo, realizado por Rosipal e Trejo em [46] sob o nome de KBPLS[7]. Com relação à versão implementada nesta tese, o KBPLS realiza o caso geral de regressão PLS2. Para tanto, os autores empregam uma reformulação do algoritmo NIPALS que, na convergência, em vez de fornecer w_i, fornece os escores t_i. Estes permitem sua expressão explícita, apesar da formulação baseada em funções de núcleo. O algoritmo NIPALS-PLS é mostrado na figura 5.2.

Uma vez calculado t_i, o algoritmo KBPLS segue calculando os coeficientes de regressão b e os resíduos. O cálculo residual é reformulado de forma semelhante à apresentada para o LPLS da seção 5.2.

Para a predição Y' o KBPLS usa uma formulação não iterativa, dada pela fórmula

$$\mathsf{Y}' = \mathsf{K}'\mathsf{U}(\mathsf{T}^\top\mathsf{K}\mathsf{U})^{-1}\mathsf{T}^\top\mathsf{Y}$$

[6] *kernel trick*.
[7] Kernel Based PLS.

Algoritmo NIPALS do KBPLS	
1	inicializar u_i de forma aleatória
2	repetir
3	$\quad \mathsf{t}_i = \mathsf{K}\mathsf{u}_i$
4	$\quad \mathsf{t}_i = \mathsf{t}_i / \|\mathsf{t}_i\|$
5	$\quad \mathsf{c} = \mathsf{Y}_i^\top \mathsf{t}_i$
6	$\quad \mathsf{u}_i = \mathsf{Y}_i \mathsf{c}$
7	$\quad \mathsf{u}_i = \mathsf{u}_i / \|\mathsf{u}_i\|$
8	até convergência de t_i

Figura 5.2: Reformulação KBPLS para o algoritmo NIPALS.

onde U e T correspondem às matrizes com os componentes u_i e t_i em suas colunas respectivamente, e K' é a matriz núcleo referente às amostras de teste. Repare que o número de fatores a serem usados para a predição é igual ao número de colunas das matrizes U e T.

Apesar de compacta, a fórmula depende da inversão da matriz $\mathsf{T}^\top\mathsf{K}\mathsf{U}$. Nos experimentos realizados, foi observada instabilidade numérica a partir de um certo número de fatores, devido à singularidade do resíduo. Uma formulação iterativa não sofre deste problema, como apresentado para o LPLS, mais adiante.

5.1.5 KPLS2

Outra forma de se obter o modelo baseado em núcleos para a regressão PLS2, é descrita a seguir para o KPLS2[8]. As formulações apresentadas são usadas para o LPLS, versão PLS1 implementada para esta tese.

Sejam X e Y as matrizes de dimensão $n \times m$ e $n \times l$, respectivamente, e $\mathbf{\Phi}$ a matriz das amostras x_i na sua nova representação no espaço $\mathcal{F}$ através do mapeamento $\Phi : \mathsf{x}_i \in \mathcal{X} \mapsto \Phi(\mathsf{x}_i) \in \mathcal{F}$ com $\mathcal{X} \subseteq \mathbb{R}^m$. Desta forma, $\mathbf{\Phi}$ possui dimensões $n \times M$, sendo que a linha i corresponde ao vetor $\Phi(\mathsf{x}_i)$.

Seja $\mathsf{C} = \mathsf{X}^\top\mathsf{Y}\mathsf{Y}^\top\mathsf{X}$ a matriz da qual queremos o autovetor w.
Observe que as matrizes X e Y são definidas por

$$\mathsf{X} = \begin{bmatrix} \mathsf{x}_1^\top \\ \vdots \\ \mathsf{x}_n^\top \end{bmatrix} \quad \text{e} \quad \mathsf{Y} = \begin{bmatrix} \mathsf{y}_1^\top \\ \vdots \\ \mathsf{y}_n^\top \end{bmatrix}$$

[8]Kernel PLS2

Portanto,

$$\mathsf{X}^\top\mathsf{Y} = [\mathsf{x}_1, \ldots, \mathsf{x}_n] \cdot \begin{bmatrix} \mathsf{y}_1^\top \\ \vdots \\ \mathsf{y}_n^\top \end{bmatrix} = \sum_{i=1}^{n} \mathsf{x}_i \mathsf{y}_i^\top$$

Seguindo uma abordagem semelhante à de Schölkopf [47] para o cálculo de componentes principais baseados em funções de núcleo, temos no espaço $\mathcal{X}$

$$\mathsf{C} = \sum_{i=1}^{n} \mathsf{x}_i \mathsf{y}_i^\top \sum_{j=1}^{n} \mathsf{y}_j \mathsf{x}_j^\top$$

que, multiplicando por w, fornece

$$\begin{aligned} \mathsf{C}\mathsf{w} &= \sum_{i=1}^{n} \mathsf{x}_i \mathsf{y}_i^\top \sum_{j=1}^{n} \mathsf{y}_j \mathsf{x}_j^\top \mathsf{w} \\ &= \sum_{i=1}^{n} \mathsf{x}_i \sum_{j=1}^{n} \mathsf{y}_i^\top \mathsf{y}_j \mathsf{x}_j^\top \mathsf{w} \end{aligned}$$

mostrando que w pertence ao sub-espaço gerado pelas amostras x_i já que

$$\mathsf{C}\mathsf{w} = \lambda\mathsf{w} \tag{5-6}$$

No espaço de características $\mathcal{F}$, temos para C

$$\begin{aligned} \mathsf{C} &= \mathbf{\Phi}^\top \mathsf{Y}\mathsf{Y}^\top \mathbf{\Phi} \\ &= \sum_{i=1}^{n} \Phi(\mathsf{x}_i) \mathsf{y}_i^\top \sum_{j=1}^{n} \mathsf{y}_j \Phi(\mathsf{x}_j)^\top \end{aligned}$$

e portanto sabemos que w está contido no sub-espaço gerado pelas amostras $\Phi(\mathsf{x}_i)$. Assim, para determinar w, basta calcular suas projeções sobre os vetores $\Phi(\mathsf{x}_i)$ para $i = 1, \ldots, n$. Isto corresponde a

$$\Phi(\mathsf{x}_k)^\top \mathsf{C}\mathsf{w} = \lambda \Phi(\mathsf{x}_k)^\top \mathsf{w} \tag{5-7}$$

para $k = 1, \ldots, n$ com

$$\mathsf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathsf{x}_i) \tag{5-8}$$

e $\alpha_i \in \mathbb{R}$.

Combinando o lado direito de (5-7) com (5-8) temos

$$\begin{aligned}\lambda\Phi(\mathsf{x}_k)^\top\mathsf{w} &= \lambda\Phi(\mathsf{x}_k)^\top\sum_{i=1}^{n}\alpha_i\Phi(\mathsf{x}_i)\\ &= \lambda\sum_{i=1}^{n}\alpha_i\Phi(\mathsf{x}_k)^\top\Phi(\mathsf{x}_i)\\ &= \lambda\sum_{i=1}^{n}\alpha_i\mathsf{K}_{k,i}\end{aligned} \tag{5-9}$$

para $k = 1, \ldots, n$ onde K é a matriz núcleo $\mathbf{\Phi}\mathbf{\Phi}^\top$.
Combinando o lado esquerdo de (5-7) com (5-8) temos

$$\begin{aligned}\Phi(\mathsf{x}_k)^\top\mathsf{C}\mathsf{w} &= \Phi(\mathsf{x}_k)^\top\sum_{i=1}^{n}\Phi(\mathsf{x}_i)\mathsf{y}_i^\top\sum_{j=1}^{n}\mathsf{y}_j\Phi(\mathsf{x}_j)^\top\sum_{u=1}^{n}\alpha_u\Phi(\mathsf{x}_u)\\ &= \sum_{i=1}^{n}\Phi(\mathsf{x}_k)^\top\Phi(\mathsf{x}_i)\sum_{j=1}^{n}\mathsf{y}_i^\top\mathsf{y}_j\sum_{u=1}^{n}\Phi(\mathsf{x}_j)^\top\Phi(\mathsf{x}_u)\alpha_u\\ &= \sum_{i=1}^{n}\mathsf{K}_{k,i}\sum_{j=1}^{n}\mathsf{y}_i^\top\mathsf{y}_j\sum_{u=1}^{n}\mathsf{K}_{j,u}\alpha_u\end{aligned} \tag{5-10}$$

Representando de forma conjunta (5-9) e (5-10) para $k = 1, \ldots, n$ obtemos a expressão matricial

$$\mathsf{K}\mathsf{Y}\mathsf{Y}^\top\mathsf{K}\alpha = \lambda\mathsf{K}\alpha \quad , \ \alpha^\top\mathsf{K}\alpha = 1 \tag{5-11}$$

onde α é o vetor contendo os coeficientes α_i, e $\alpha^\top\mathsf{K}\alpha = 1$ corresponde a $\|\mathsf{w}\| = 1$.

Existem vários algoritmos para resolver a equação (5-11) como mostrado em [11] para o caso geral $\mathsf{A}\mathsf{x} = \lambda\mathsf{B}\mathsf{x}$ onde A e B são Hermitianas.

Com relação ao trabalho de Rosipal e Trejo [46], a resolução da equação (5-11) permite o cálculo de w e conseqüentemente de t, sem que este último seja normalizado. Em [46], utilizando o fato de t ser autovetor da matriz $\mathsf{X}\mathsf{X}^\top\mathsf{Y}\mathsf{Y}^\top$, é fornecida uma versão baseada em núcleos para o NIPALS, que na convergência fornece o componente t normalizado:

$$\mathsf{K}\mathsf{Y}\mathsf{Y}^\top\mathsf{t} = \lambda\mathsf{t} \quad \text{com } \|\mathsf{t}\| = 1$$

Isto equivale a resolver (5-11) com $\mathsf{t} = \mathsf{K}\alpha$ impondo $\|\mathsf{t}\| = \alpha^\top\mathsf{K}^2\alpha = 1$.

Na versão original do algoritmo PLS [56, 57], é calculado via NIPALS o vetor w, autovetor de $\mathsf{X}^\top\mathsf{Y}\mathsf{Y}^\top\mathsf{X}$ associado ao mesmo autovalor. Isto torna possível que os escores t permaneçam não normalizados, permitindo a

análise da variável latente escolhida.

O cálculo dos escores, cargas e resíduos pode ser realizado de forma semelhante à apresentada no LPLS. A principal diferença nas formulações PLS1 e PLS2 está no cálculo de $\mathbf{w}$, resolvido nesta seção, e no cálculo residual de $\mathbf{Y}$ que não apresenta alterações nas versões baseadas em núcleos.

5.2 LPLS

Rosipal e Trejo em [46] apresentam uma versão geral para o PLS2, onde funções de núcleo são usadas para mapear as amostras originais num espaço de maior dimensão e em seguida o algoritmo de regressão linear por mínimos quadrados parciais é aplicado. LPLS[9] é um algoritmo para regressão PLS1. Em relação a [46], a construção do algoritmo LPLS para o caso particular de apenas uma variável dependente permite não somente uma formulação mais simples, mas também maior eficiência de desempenho.

Para a elaboração do LPLS é necessária tanto a reformulação do algoritmo de regressão para cálculo da estrutura latente, quanto a do algoritmo de predição para seleção de fatores. Um dos aspectos da versão baseada em núcleos é a expressão de $\mathbf{w}$, autovetor de $\mathbf{X}^\top\mathbf{Y}\mathbf{Y}^\top\mathbf{X}$, no espaço $\mathcal{F}$. Não é recomendado, e muitas vezes impossível, exprimir explicitamente este vetor pois sua dimensão depende da dimensão do espaço $\mathcal{F}$. Logo o algoritmo LPLS não possui uma expressão explícita para $\mathbf{w}$, mas apenas para os escores $\mathbf{t}$ cuja dimensão corresponde ao número de amostras.

5.2.1 A estrutura latente

A formulação com funções de núcleo é apresentada para conjuntos de fatores calculados a cada iteração do algoritmo de regressão PLS1.

[9] Lifted PLS.

Cálculo de w

A regressão PLS1 possui apenas uma variável dependente. Neste caso, (5-11) aceita a solução trivial $\alpha = \mathsf{Y}$, o que substituindo em (5-8) fornece

$$\begin{aligned}\mathsf{w} &= \sum_{i=1}^{n} \Phi(\mathsf{x}_i)\mathsf{y}_i \\ &= \mathbf{\Phi}^\top \mathsf{Y}\end{aligned}$$

sendo que $\|\mathsf{w}\|^2 = \mathsf{w}^\top\mathsf{w} = \mathsf{Y}^\top\mathbf{\Phi}\mathbf{\Phi}^\top\mathsf{Y} = \mathsf{Y}^\top\mathsf{K}\mathsf{Y}$, fornecendo para o vetor normalizado

$$\mathsf{w} = \mathbf{\Phi}^\top\mathsf{Y}/(\mathsf{Y}^\top\mathsf{K}\mathsf{Y})^{1/2} \tag{5-12}$$

Cálculo dos escores, cargas e coeficiente de regressão

Com relação ao algoritmo PLS1 apresentado na figura 2.2, possuímos para o LPLS apenas a expressão do autovetor, correspondente às linhas 4 e 5. Para o cálculo dos escores t no espaço $\mathcal{F}$, temos com o uso da equação (5-12)

$$\begin{aligned}\mathsf{t} &= \mathbf{\Phi}\mathsf{w} \\ &= \mathbf{\Phi}\mathbf{\Phi}^\top\mathsf{Y}/(\mathsf{Y}^\top\mathsf{K}\mathsf{Y})^{1/2} \\ &= \mathsf{K}\mathsf{Y}/(\mathsf{Y}^\top\mathsf{K}\mathsf{Y})^{1/2}\end{aligned}$$

O coeficiente de regressão linear b não sofre modificações, pois depende apenas de t e Y. Para as cargas p, temos

$$\mathsf{p} = \mathbf{\Phi}^\top\mathsf{t}/\mathsf{t}^\top\mathsf{t} \tag{5-13}$$

Como já notado, apesar de possuirmos expressões para $\mathsf{w}, \mathsf{t}, b$ e p, podemos calcular explicitamente apenas t e b, o que não é um problema pois são suficientes para o modelo procurado.

As expressões dos fatores até agora apresentados correspondem a uma iteração do algoritmo PLS1. Para as demais, é necessário realizar o cálculo residual de X e Y, que novamente não é possível de forma explícita no espaço $\mathcal{F}$. Para tanto, iremos realizar o cálculo residual da matriz núcleo K, que além de possuir expressão calculável, é suficiente para o cálculo do modelo através de t e b.

Cálculo residual de K

No espaço de entrada $\mathcal{X}$ temos entre as iterações i e $i+1$ do algoritmos PLS1

$$\mathsf{X}_{i+1} = \mathsf{X}_i - \mathsf{t}_i\mathsf{p}_i^\top$$

que em $\mathcal{F}$ possui a expressão

$$\boldsymbol{\Phi}_{i+1} = \boldsymbol{\Phi}_i - \mathsf{t}_i\mathsf{p}_i^\top$$

Temos então para a matriz núcleo K_{i+1}

$$\begin{aligned}\mathsf{K}_{i+1} &= \boldsymbol{\Phi}_{i+1}\boldsymbol{\Phi}_{i+1}^\top \\ &= (\boldsymbol{\Phi}_i - \mathsf{t}_i\mathsf{p}_i^\top)(\boldsymbol{\Phi}_i^\top - \mathsf{p}_i\mathsf{t}_i^\top)\end{aligned}$$

que juntamente com (5-13) resulta

$$\mathsf{K}_{i+1} = \left(\boldsymbol{\Phi}_i - \frac{\mathsf{t}_i\mathsf{t}_i^\top\boldsymbol{\Phi}}{\mathsf{t}_i^\top\mathsf{t}_i}\right)\left(\boldsymbol{\Phi}_i^\top - \frac{\boldsymbol{\Phi}^\top\mathsf{t}_i\mathsf{t}_i^\top}{\mathsf{t}_i^\top\mathsf{t}_i}\right)$$

definindo $a_i = \mathsf{t}_i^\top\mathsf{t}_i$, obtemos

$$\mathsf{K}_{i+1} = \left(\boldsymbol{\Phi}_i - \frac{\mathsf{t}_i\mathsf{t}_i^\top\boldsymbol{\Phi}}{a_i}\right)\left(\boldsymbol{\Phi}_i^\top - \frac{\boldsymbol{\Phi}^\top\mathsf{t}_i\mathsf{t}_i^\top}{a_i}\right)$$

que desenvolvendo fornece

$$\begin{aligned}\mathsf{K}_{i+1} &= \boldsymbol{\Phi}_i\boldsymbol{\Phi}_i^\top - \frac{\boldsymbol{\Phi}_i\boldsymbol{\Phi}^\top\mathsf{t}_i\mathsf{t}_i^\top}{a_i} - \frac{\mathsf{t}_i\mathsf{t}_i^\top\boldsymbol{\Phi}\boldsymbol{\Phi}_i^\top}{a_i} + \frac{\mathsf{t}_i\mathsf{t}_i^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\mathsf{t}_i\mathsf{t}_i^\top}{a_i^2} \\ &= \mathsf{K}_i - \frac{\mathsf{K}_i\mathsf{t}_i\mathsf{t}_i^\top}{a_i} - \frac{\mathsf{t}_i\mathsf{t}_i^\top\mathsf{K}_i}{a_i} + \frac{\mathsf{t}_i\mathsf{t}_i^\top\mathsf{K}_i\mathsf{t}_i\mathsf{t}_i^\top}{a_i^2}\end{aligned}$$

Para a garantia de uma implementação mais eficiente vale definir $\mathsf{g}_i = \mathsf{K}_i\mathsf{t}_i$, sendo que $\mathsf{g}_i^\top = \mathsf{t}_i^\top\mathsf{K}_i$, já que K_i é simétrica. Com isto, temos

$$\begin{aligned}\mathsf{K}_{i+1} &= \mathsf{K}_i - \frac{\mathsf{g}_i\mathsf{t}_i^\top}{a_i} - \frac{\mathsf{t}_i\mathsf{g}_i^\top}{a_i} + \frac{\mathsf{t}_i\mathsf{t}_i^\top\mathsf{g}_i\mathsf{t}_i^\top}{a_i^2} \\ &= \mathsf{K}_i - \mathsf{U} - \mathsf{U}^\top + \frac{(\mathsf{t}_i^\top\mathsf{g}_i)\mathsf{t}_i\mathsf{t}_i^\top}{a_i^2}\end{aligned}$$

com $\mathsf{U} = \dfrac{\mathsf{g}_i\mathsf{t}_i^\top}{a_i}$.

Desta forma, possuímos uma fórmula para a atualização da matriz de núcleo resultando no algoritmo LPLS apresentado na figura 5.3.

Algoritmo para regressão PLS1 baseado em funções de núcleo		
1	construir K com a função de núcleo desejada	
2	Y_1	$\leftarrow \mathsf{Y}$
3	for	$\mathtt{i} = 1$ to $\mathtt{k}$
4		// norma ao quadrado do autovetor
5		$h_i \leftarrow \mathsf{Y}_i^\top \mathsf{K}\mathsf{Y}_i$
6		$\mathsf{t}_i \leftarrow \mathsf{K}\mathsf{Y}_i / \sqrt{h_i}$
		// Cálculo do coeficiente de regressão
7		$a_i \leftarrow \mathsf{t}_i^\top \mathsf{t}_i$
8		$b_i \leftarrow \mathsf{Y}_i^\top \mathsf{t}_i / a_i$
		// Cálculo residual
9		$\mathsf{g}_i \leftarrow \mathsf{K}\mathsf{t}_i$
10		$\mathsf{U} \leftarrow \mathsf{g}_i \mathsf{t}_i^\top / a_i$
11		$\mathsf{K} \leftarrow \mathsf{K} - \mathsf{U} - \mathsf{U}^\top + (\mathsf{t}_i^\top \mathsf{g}_i)\mathsf{t}_i \mathsf{t}_i^\top / a_i^2$
12		$\mathsf{Y}_{i+1} \leftarrow \mathsf{Y}_i - b_i \mathsf{t}_i$
13	end	

Figura 5.3: Algoritmo LPLS.

5.2.2 Predição

É necessário também reformular o algoritmo para predição PLS1 considerando funções de núcleo. Com relação ao algoritmo apresentado na figura 2.3, temos para os escores no espaço $\mathcal{X}$

$$\mathsf{t}' = \mathsf{X}'\mathsf{w}$$

onde X' corresponde à matriz com o conjunto independente de amostras para teste e w o autovetor calculado na fase de treinamento. Desta forma, obtemos em $\mathcal{F}$

$$\mathsf{t}' = \mathbf{\Phi}'\mathsf{w}$$

onde $\mathbf{\Phi}'$ corresponde à matriz com a nova representação das amostras de teste, obtida com o mesmo mapeamento Φ usado com as amostras de treino. Substituindo com a equação (5-12), obtemos

$$\begin{aligned} \mathsf{t}' &= \mathbf{\Phi}'\mathbf{\Phi}^\top \mathsf{Y} / (\mathsf{Y}^\top \mathsf{K}\mathsf{Y})^{1/2} \\ &= \mathsf{K}'\mathsf{Y} / (\mathsf{Y}^\top \mathsf{K}\mathsf{Y})^{1/2} \end{aligned}$$

sendo $\mathsf{K}' = \mathbf{\Phi}'\mathbf{\Phi}^\top$ a matriz núcleo de teste.

Como ocorreu na fase de treinamento, o cálculo residual é realizado na

matriz K' que pode ser calculada explicitamente. De acordo com o algoritmo de predição PLS temos

$$\mathsf{X}'_{i+1} = \mathsf{X}'_i - \mathsf{t}'_i\mathsf{p}_i^\top$$

que em $\mathcal{F}$ resulta

$$\mathbf{\Phi}'_{i+1} = \mathbf{\Phi}'_i - \mathsf{t}'_i\mathsf{p}_i^\top$$

Logo, para a matriz núcleo de teste, temos

$$\begin{aligned}\mathsf{K}'_{i+1} &= \mathbf{\Phi}'_{i+1}\mathbf{\Phi}_{i+1}^\top \\ &= (\mathbf{\Phi}'_i - \mathsf{t}'_i\mathsf{p}_i^\top)(\mathbf{\Phi}_i^\top - \mathsf{p}_i\mathsf{t}_i^\top)\end{aligned}$$

substituindo p pela equação 5-13

$$\begin{aligned}&= \left(\mathbf{\Phi}'_i - \mathsf{t}'_i\frac{\mathsf{t}_i^\top\mathbf{\Phi}_i}{a_i}\right)\left(\mathbf{\Phi}_i^\top - \frac{\mathbf{\Phi}_i^\top\mathsf{t}_i}{a_i}\mathsf{t}_i^\top\right) \\ &= \mathbf{\Phi}'_i\mathbf{\Phi}_i^\top - \frac{\mathbf{\Phi}'_i\mathbf{\Phi}_i^\top\mathsf{t}_i\mathsf{t}_i^\top}{a_i} - \frac{\mathsf{t}'_i\mathsf{t}_i^\top\mathbf{\Phi}_i\mathbf{\Phi}_i^\top}{a_i} + \frac{\mathsf{t}'_i\mathsf{t}_i^\top\mathbf{\Phi}_i\mathbf{\Phi}_i^\top\mathsf{t}_i\mathsf{t}_i^\top}{a_i^2} \\ &= \mathsf{K}'_i - \frac{\mathsf{K}'_i\mathsf{t}_i\mathsf{t}_i^\top}{a_i} - \frac{\mathsf{t}'_i\mathsf{t}_i^\top\mathsf{K}_i}{a_i} + \frac{\mathsf{t}'_i\mathsf{t}_i^\top\mathsf{K}_i\mathsf{t}_i\mathsf{t}_i^\top}{a_i^2} \\ &= \mathsf{K}'_i - \frac{\mathsf{K}'_i\mathsf{t}_i\mathsf{t}_i^\top}{a_i} - \frac{\mathsf{t}'_i\mathsf{g}_i^\top}{a_i} + \frac{\mathsf{t}'_i\mathsf{t}_i^\top\mathsf{g}_i\mathsf{t}_i^\top}{a_i^2} \\ &= \mathsf{K}'_i - \frac{\mathsf{K}'_i\mathsf{t}_i\mathsf{t}_i^\top}{a_i} - \frac{\mathsf{t}'_i\mathsf{g}_i^\top}{a_i} + \frac{(\mathsf{t}_i^\top\mathsf{g}_i)\mathsf{t}'_i\mathsf{t}_i^\top}{a_i^2}\end{aligned}$$

O algoritmo para predição LPLS é mostrado na figura 5.4 Repare que para a

Algoritmo para predição PLS1 baseado em funções de núcleo	
1	construir K' com a mesma função de núcleo
	usada para treinamento
2	$\mathsf{Y}' \leftarrow 0$
3	for $\mathtt{i} = 1$ to $\mathtt{k}$
4	$\mathsf{t}'_i \leftarrow \mathsf{K}'\mathsf{Y}_i/\sqrt{h_i}$
5	$\mathsf{Y}' \leftarrow \mathsf{Y}' + b_i\mathsf{t}'_i$
	// Cálculo residual
6	$\mathsf{K}' = \mathsf{K}' - \mathsf{K}'\mathsf{t}_i\mathsf{t}_i^\top/a_i - \mathsf{t}'_i\mathsf{g}_i^\top/a_i + (\mathsf{t}_i^\top\mathsf{g}_i)\mathsf{t}'_i\mathsf{t}_i^\top/a_i^2$
7	end

Figura 5.4: Predição LPLS.

predição são necessários os seguintes dados obtidos durante o treinamento:

1. $\{\mathsf{t}_i, b_i, \mathsf{g}_i, a_i, h_i\}$ que compõem o modelo de regressão, sendo que g_i e a_i são usados para eficiência na predição;

2. os resíduos Y_i.

O algoritmo de predição é necessário para o cálculo do PRESS para a seleção do número de fatores. Com relação ao treinamento do modelo, o LPLS requer além do número de fatores a serem usados, a determinação do núcleo e de seus parâmetros, tornando a tarefa de aprendizagem um pouco mais complexa. De qualquer forma, os resultados obtidos com o exemplo mostrado a seguir e os já reportados em [46], comprovam a validade do método.

5.2.3 Exemplo

Os experimentos em [46] comprovam que a modelagem via funções de núcleo é fundamental para a melhora da qualidade de predição em certos conjuntos de dados. O exemplo realizado tem por objetivo ilustrar o poder da modelagem via funções de núcleo, justificando seu uso nos algoritmos das próximas seções.

Com relação à formulação PLS2 baseada em funções de núcleo apresentada em [46], o algoritmo LPLS, apesar de restrito a uma variável dependente, mostrou não somente maior eficiência computacional mas maior estabilidade numérica. Em [46], a predição depende da inversão de uma matriz, o que a partir de um certo número de fatores gera instabilidade numérica devido à singularidade da matriz residual. Já a abordagem de LPLS, sendo iterativa, fornece resultados mais estáveis.

Para mostrar o ganho obtido com a modelagem LPLS, foi empregado *Meat* do conjunto NIR. É usada a função de núcleo polinomial (5-4) para a construção da matriz núcleo tanto de treino quanto de teste. Para ambas, o grau escolhido para o polinômio é 2, sendo a constante igual a 1. Na figura 5.5 encontramos a curva PRESS representativa de uma das 20 amostragens realizadas com o conjunto *Meat*. Na média obtivemos um ganho de 50% na predição, evidenciando a importância da modelagem com funções de núcleo. Um detalhe observado com o uso de núcleos polinomiais é a incapacidade de obter uma boa predição da variável dependente Y com poucos fatores, o que leva à procura de novas formulações não-lineares para o PLS.

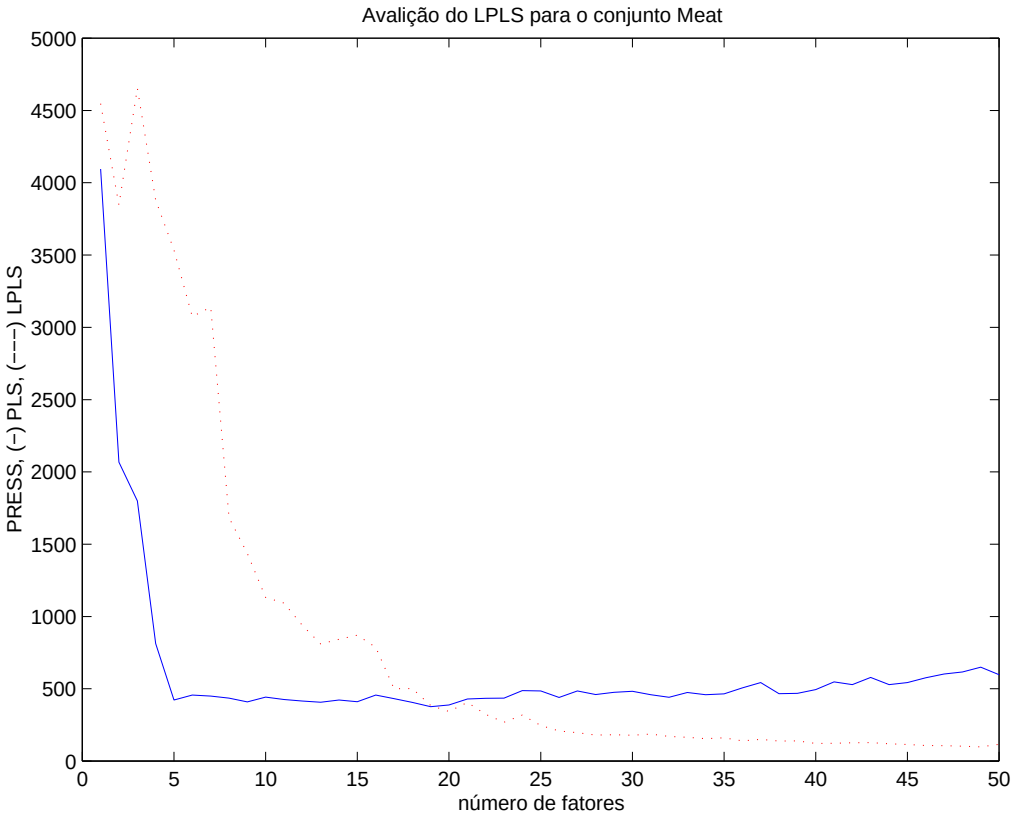

Figura 5.5: Comparação do LPLS com o PLS.

5.3 KDPLS

O algoritmo DPLS fornece uma aproximação para a regressão PLS no caso de mais de uma variável dependente. É razoável que os resultados encontrados com a formulação linear se repitam com uma abordagem baseada em núcleos, ou seja, boa aproximação na predição e ganho significativo no desempenho computacional. Isto leva à elaboração de KDPLS[10].

5.3.1 Formulação não linear para a aproximação

Como mencionado com o LPLS, é necessário reformular o DPLS de forma que as amostram estejam envolvidas apenas em cálculo de produto interno, permitindo que a função de núcleo seja introduzida. Com relação ao LPLS, a única modificação consiste no cálculo do autovetor aproximado. No espaço de entrada $\mathcal{X}$ temos para os autovetores e autovalores das matrizes

[10] Kernel Direct PLS.

de posto 1

$$\mathbf{w}_i = \mathbf{X}^\top \mathbf{Y}_i$$

e

$$\lambda_i = \mathbf{w}_i^\top \mathbf{w}_i$$

para $i = 1, \ldots, l$. No espaço das características $\mathcal{F}$, isto corresponde a

$$\mathbf{w}_i = \mathbf{\Phi}^\top \mathbf{Y}_i$$

e

$$\begin{aligned}\lambda_i &= \mathbf{w}_i^\top \mathbf{w}_i \\ &= \mathbf{Y}_i^\top \mathbf{\Phi}\mathbf{\Phi}^\top \mathbf{Y}_i \\ &= \mathbf{Y}_i^\top \mathbf{K}_i \mathbf{Y}_i\end{aligned}$$

para $i = 1, \ldots, l$. Com isto, podemos determinar o autovetor $\mathbf{w}_{(1)}$ associado ao maior autovalor $\lambda_{(1)}$. Desta forma, a aproximação em $\mathcal{F}$ corresponde a

$$\begin{aligned}\mathbf{w} &= \sum_{i=1}^{l} \lambda_i \mathbf{w}_i (\mathbf{w}_i^\top \mathbf{w}_{(1)}) \\ &= \sum_{i=1}^{l} \lambda_i \mathbf{\Phi}^\top \mathbf{Y}_i (\mathbf{Y}_i^\top \mathbf{\Phi}\mathbf{\Phi}^\top \mathbf{Y}_{(1)}) \\ &= \sum_{i=1}^{l} \lambda_i \mathbf{\Phi}^\top \mathbf{Y}_i (\mathbf{Y}_i^\top \mathbf{K}\mathbf{Y}_{(1)}) \\ &= \mathbf{\Phi}^\top \sum_{i=1}^{l} \lambda_i \mathbf{Y}_i (\mathbf{Y}_i^\top \mathbf{K}\mathbf{Y}_{(1)})\end{aligned}$$

e definindo $\mathbf{v} = \sum_{i=1}^{l} \lambda_i \mathbf{Y}_i (\mathbf{Y}_i^\top \mathbf{K}\mathbf{Y}_{(1)})$, temos

$$\mathbf{w} = \mathbf{\Phi}^\top \mathbf{v}$$

cuja norma ao quadrado é dada por

$$\begin{aligned}\|\mathsf{w}\|^2 &= \mathsf{w}^\top \mathsf{w} \\ &= \mathsf{v}^\top \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \mathsf{v} \\ &= \mathsf{v}^\top \mathsf{K}\mathsf{v}\end{aligned}$$

com isto, temos para o autovetor aproximado normalizado

$$\mathsf{w} = \frac{\boldsymbol{\Phi}^\top \mathsf{v}}{(\mathsf{v}^\top \mathsf{K}\mathsf{v})^{\frac{1}{2}}}$$

Sabemos que w não pode ser explicitamente calculado, mas podemos calcular os escores obtendo

$$\begin{aligned}\mathsf{t} &= \boldsymbol{\Phi}\mathsf{w} \\ &= \frac{\boldsymbol{\Phi}\boldsymbol{\Phi}^\top \mathsf{v}}{(\mathsf{v}^\top \mathsf{K}\mathsf{v})^{\frac{1}{2}}} \\ &= \frac{\mathsf{K}\mathsf{v}}{(\mathsf{v}^\top \mathsf{K}\mathsf{v})^{\frac{1}{2}}}\end{aligned}$$

Para o restante dos fatores e o cálculo residual, as equações são as mesmas do que para o LPLS.

Porém é necessário um pequeno ajuste para o algoritmo de predição no cálculo do escores t'

$$\begin{aligned}\mathsf{t}' &= \boldsymbol{\Phi}'\mathsf{w} \\ &= \frac{\boldsymbol{\Phi}'\boldsymbol{\Phi}^\top \mathsf{v}}{(\mathsf{v}^\top \mathsf{K}\mathsf{v})^{\frac{1}{2}}} \\ &= \frac{\mathsf{K}'\mathsf{v}}{(\mathsf{v}^\top \mathsf{K}\mathsf{v})^{\frac{1}{2}}}\end{aligned}$$

5.3.2 Resultados experimentais

Para comprovar seu desempenho, o KDPLS é avaliado em 5 conjuntos de dados do conjunto NIR, são eles: *Wheat, Light gas oil, Corn, Polymer* e *Combustible*. A base para a implementação foi o LPLS, sendo este adaptado para mais de uma variável dependente: b passando a ser um vetor de coeficientes de regressão por mínimos quadrados. Para comparação é usado o algoritmo PLS2 baseado em funções de núcleo descrito em [46], chamado

de KBPLS[11]. Duas funções de núcleo são usadas para a modelagem aproximada: a função polinomial de grau 2 com constante 1 e a gaussiana com desvio 3,1. Outros valores poderiam ser usados com as funções, mas como o objetivo é mostrar a viabilidade do algoritmo KDPLS, foram realizados experimentos apenas com os parâmetros mais usados nos experimentos desta tese.

Na tabela 5.1, são apresentados os resultados experimentais com a função de núcleo polinomial, dadas as mesmas métricas usadas para a avaliação do DPLS. Da mesma forma, na tabela 5.2 encontramos os resultados com a função de núcleo gaussiana.

Tabela 5.1: Comparação entre KBPLS e KDPLS com núcleo polinomial

Conjunto	N. fatores		PRESS		KBPLS
	KBPLS	KDPLS	KBPLS	KDPLS	iterações
Wheat	24	28	11,17	11,58	62,3
Light gas oil	10	11	73,86	72,81	63,8
Corn	18	22	5,82	5,77	61,2
Polymer	9	9	0,60	0,58	41,0
Combustible	10	12	228,08	232,34	48,3

Como podemos observar, os resultados obtidos com o KBPLS e o KDPLS são próximos. Obtemos curvas PRESS semelhantes para os dois modelos com ambas funções de núcleo, como podemos observar nas figuras 5.6 e 5.7.

Tabela 5.2: Comparação entre KBPLS e KDPLS com núcleo gaussiano

Conjunto	N. fatores		PRESS		KBPLS
	KBPLS	KDPLS	KBPLS	KDPLS	iterações
Wheat	21	22	12,82	13,09	65,5
Light gas oil	13	14	70,79	67,58	67,2
Corn	15	20	6,10	5,81	63,0
Polymer	9	10	0,41	0,42	50,0
Combustible	9	12	236,86	234,26	52,2

Mesmo que em alguns casos o desempenho do KDPLS tenha sido ligeiramente inferior, a curva PRESS mantém o mesmo comportamento, como observado, por exemplo, para o conjunto *Polymer* com núcleo gaussiano na figura 5.8.

[11] Kernel Based PLS.

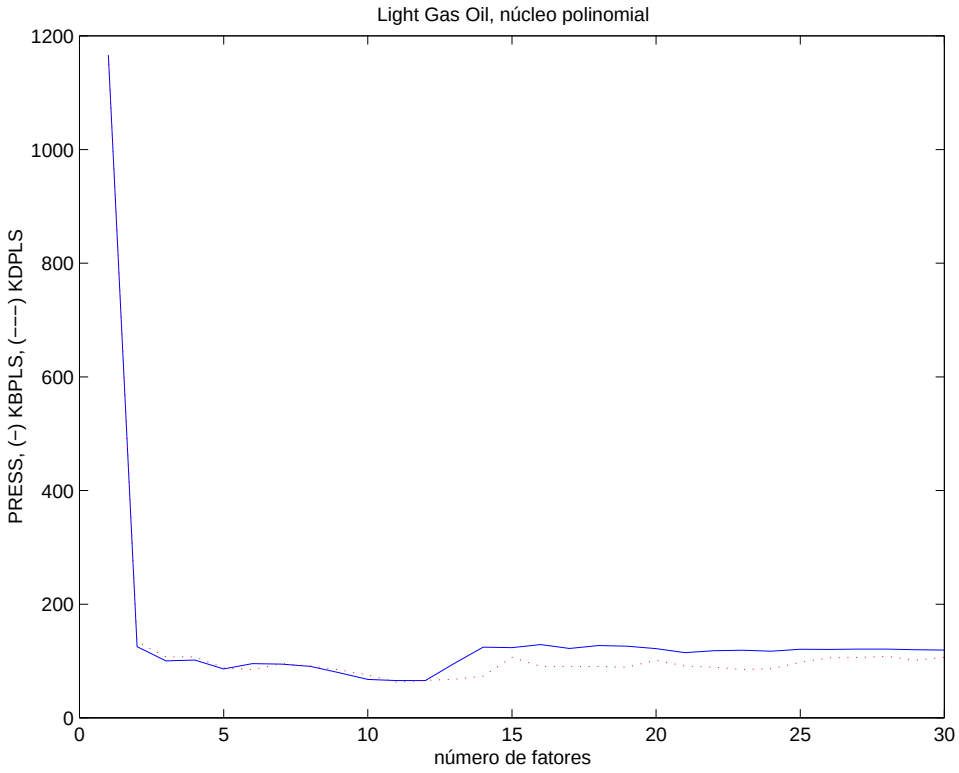

Figura 5.6: PRESS dos modelos KBPLS e KDPLS para o conjunto Light Gas Oil.

Com relação à aproximação para o caso linear, os resultados obtidos com o KDPLS revelam que para um número maior de fatores o comportamento das duas curvas PRESS tende a se diferenciar, como vemos na figura 5.9.

Cada conjunto foi sorteado 10 vezes. Na tabela foram reportadas amostragens representativas. A diferença relativa média entre o PRESS mínimo foi de -3,71% a 4,82% para todos os conjuntos. Além disto, como esperado, o desempenho computacional do KDPLS ficou acima de 40% melhor do que o do KBPLS quando calculados somente os 30 primeiros fatores. A curva PRESS não foi avaliada para mais fatores devido à instabilidade encontrada na formulação KBPLS de [46].

Os resultados obtidos comprovam a utilidade do KDPLS quando for necessária a aplicação do PLS2 baseado em funções de núcleo em grandes conjuntos de dados.

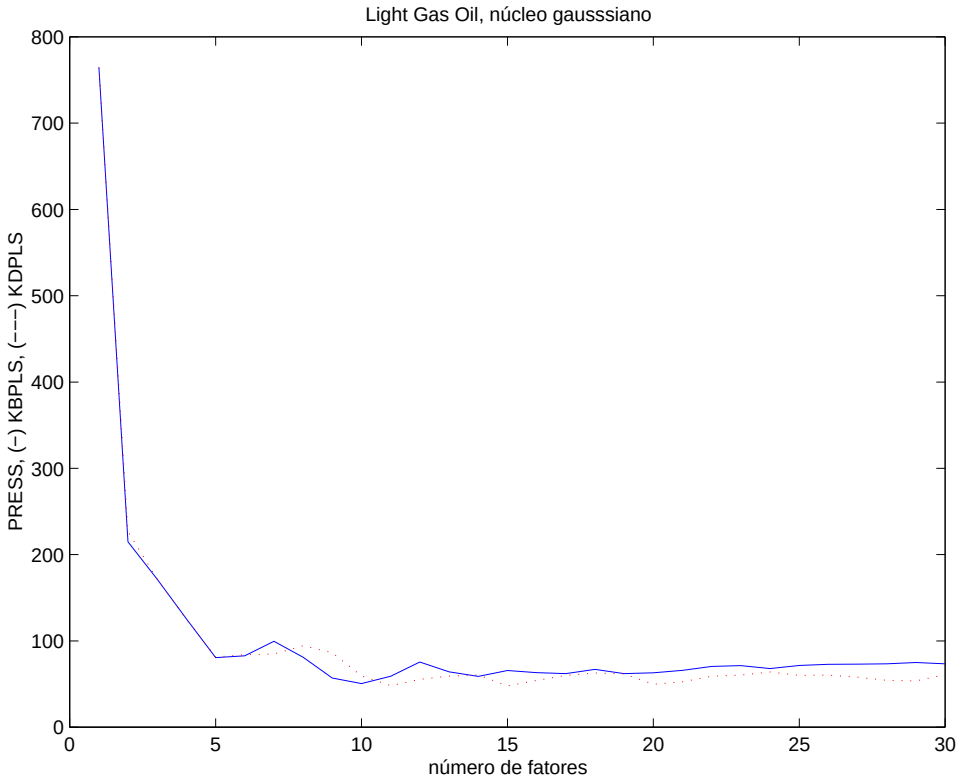

Figura 5.7: PRESS dos modelos KBPLS e KDPLS para o conjunto Light Gas Oil.

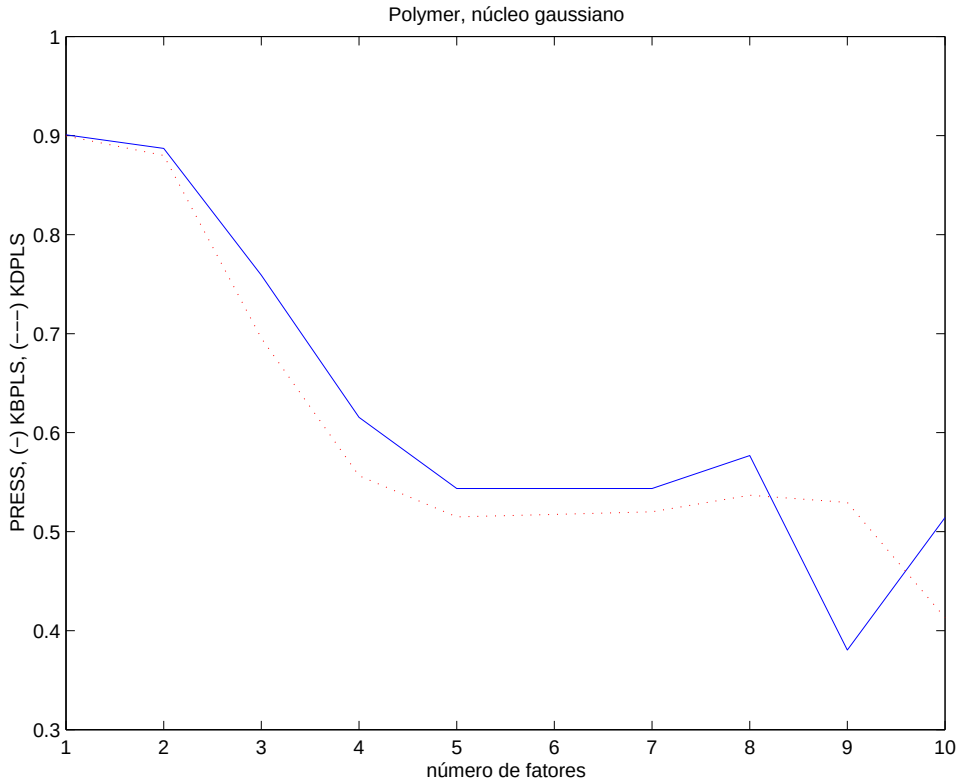

Figura 5.8: PRESS dos modelos KBPLS e KDPLS para o conjunto Polymer.

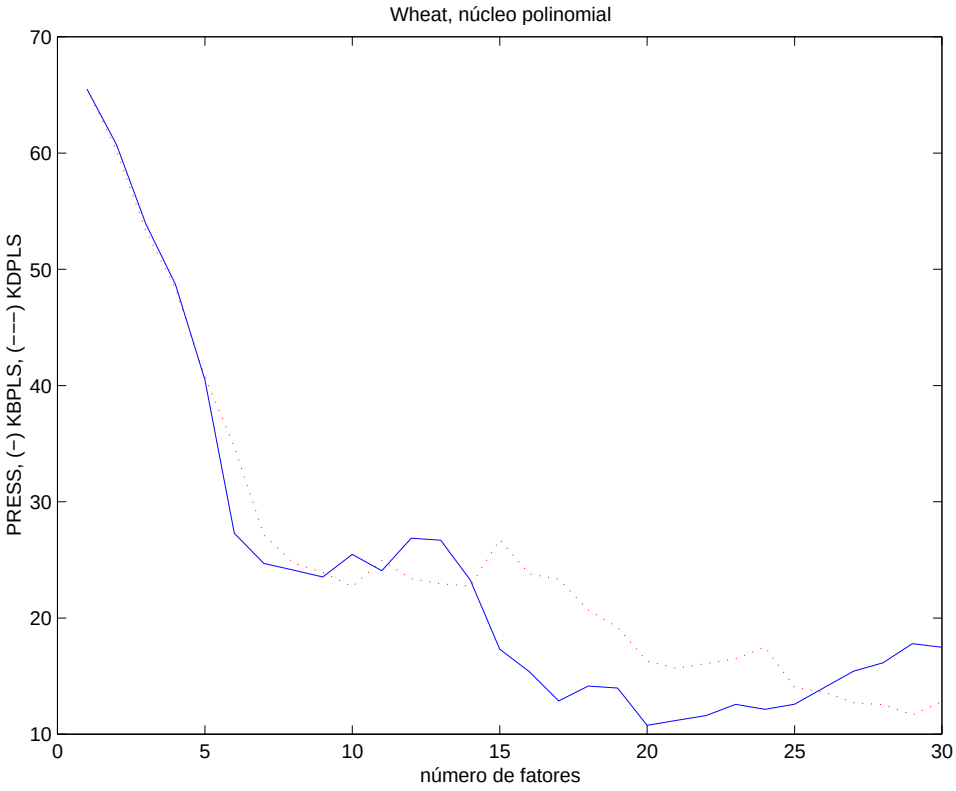

Figura 5.9: PRESS dos modelos KBPLS e KDPLS para o conjunto Wheat.

5.4 MKPLS - PLS Multi Núcleos

A principal motivação para o desenvolvimento do algoritmo MKPLS[12] está na comparação entre as curvas PRESS obtidas com o PLS baseado em uma função de núcleo e o PLS linear tradicional. Por exemplo, se observarmos o desempenho dos dois modelos com o conjunto de dados *Meat* apresentado na figura 5.5, vemos que a regressão baseada em um núcleo fornece menor erro de predição se forem usados mais de 19 fatores. Por outro lado, se um número menor de fatores for usado, a modelagem de um núcleo apresenta um desempenho muito fraco, sendo o uso do PLS linear recomendado. Para o exemplo fornecido, o núcleo polinomial usado $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j)^2$ mal consegue modelar a variável dependente $\mathbf{Y}$ para os dez primeiros fatores. Seria vantajoso o uso de um modelo de regressão capaz de fornecer tanto a mesma qualidade do PLS linear nos primeiros fatores, quanto a qualidade de predição para os demais fatores.

[12] Multi-Kernel PLS

O algoritmo MKPLS constitui uma generalização de algoritmos não-lineares PLS baseados em uma função de núcleo, como o LPLS ou KBPLS [46], pois emprega uma matriz de núcleo K_1 para o cálculo dos primeiros f_1 fatores e realiza a troca para uma matriz de núcleo diferente K_2 para os demais fatores.

5.4.1 Treinamento

Dadas duas funções de núcleo K_1 e K_2, e as respectivas matrizes núcleo K_1 e K_2 construídas com o conjunto de treino, o modelo de regressão MKPLS é calculado com os seguintes passos:

1. calcular os primeiros f_1 fatores $\{\mathsf{t}_i, \mathsf{b}_i\}$ através do algoritmo para regressão PLS não-linear baseado em um núcleo aplicado à matriz K_1;

2. descontar da segunda matriz núcleo K_2 e das saídas Y os fatores calculados usando o algoritmo para desconto apresentado na figura 5.10;

3. aplicar novamente o algoritmo PLS de um núcleo na matriz descontada K_2, obtendo os f_2 fatores restantes do modelo.

Ao final do procedimento, o conjunto de $f_1 + f_2$ fatores constitui o modelo não-linear MKPLS.

Desconto de $\mathtt{K_2}$ e $\mathtt{Y}$ para a fase de treinamento		
1	for	$\mathtt{i} = 1$ to $\mathtt{f_1}$
2		$\mathsf{g}'_i \leftarrow \mathsf{K}_2\mathsf{t}_i$
3		$a_i \leftarrow \mathsf{t}_i^\top \mathsf{t}_i$
3		$\mathsf{U} \leftarrow \mathsf{g}'_i\mathsf{t}_i^\top / a_i$
5		$\mathsf{K}_2 \leftarrow \mathsf{K}_2 - \mathsf{U} - \mathsf{U}^\top + (\mathsf{t}_i^\top \mathsf{g}'_i)\mathsf{t}_i\mathsf{t}_i^\top / a_i^2$
6		$\mathsf{Y} \leftarrow \mathsf{Y} - \mathsf{t}_i\mathsf{b}_i^\top$
7	end	

Figura 5.10: Algoritmo MKPLS para desconto no treinamento.

Notar que no algoritmo de desconto é gerado o conjunto de vetores $\{\mathsf{g}'_i\}$ com $\mathsf{g}'_i = \mathsf{K}_2\mathsf{t}_i$, projeção dos fatores encontrados com a matriz K_1 na matriz núcleo K_2. Estes vetores são necessários para o algoritmo de desconto na predição. Já os coeficientes a_i, por dependerem apenas de t_i ($a_i = \mathsf{t}_i^\top \mathsf{t}_i$), são os mesmo calculados com a matriz K_1.

5.4.2 Predição

Como a predição via funções de núcleo requer uma matriz núcleo específica para o conjunto de teste, é também necessária a troca de matrizes núcleo de K'_1 para K'_2 nesta etapa. Os seguintes passos são necessários para a predição:

1. aplicar o algoritmo de predição na matriz K'_1. Porém, o conjunto de f_1 fatores t'_i deve ser armazenado junto com a predição Y';

2. realizar o desconto da matriz de núcleo K'_2 com o algoritmo da figura 5.11. Notar o uso de g'_i obtido durante o desconto do treinamento;

3. aplicar o algoritmo de predição na matriz núcleo K'_2 descontada, sendo que a predição começa com o Y' obtido no passo 1.

	Desconto de K'_2 na fase de predição	
1	for	$\mathtt{i} = 1$ to $\mathtt{f}_1$
2		$\mathsf{U}_1 \leftarrow \mathsf{K}'_2 \mathsf{t}_i \mathsf{t}_i^\top / a_i$
3		$\mathsf{U}_2 \leftarrow \mathsf{t}'_i \mathsf{g}_i'^\top / a_i$
4		$\mathsf{K}'_2 = \mathsf{K}'_2 - \mathsf{U}_1 - \mathsf{U}_2 + (\mathsf{t}_i^\top \mathsf{g}'_i) \mathsf{t}'_i \mathsf{t}_i^\top / a_i^2$
5	end	

Figura 5.11: Algoritmo MKPLS para desconto na predição.

Num primeiro momento, o desconto na predição parece desnecessário, indicando que a predição pode ser realizada para os $(f_1 + f_2)$ fatores começando diretamente com a matriz K'_2. Isto não é verdade, pois para termos a mesma qualidade de predição conseguida com a primeira função de núcleo, é necessário descontar na predição os fatores $\{\mathsf{t}'_i\}$ que só podem ser calculados realizando a predição com a primeira matriz núcleo K'_1 para f_1 fatores.

5.4.3 O algoritmo

Do ponto de vista semântico, é importante que o mapeamento implícito com a função de núcleo K_2 esteja incluído no mapeamento da primeira função K_1. Isto se deve ao fato de a troca de núcleos ser realizada descontando os fatores t_i encontrados com K_1. Não faria sentido descontar de K_2 um fator t_i que não correspondesse a uma combinação linear das

variáveis na nova representação. Esta característica, que do ponto de vista operacional não é obrigatória, pode ser conseguida facilmente definindo a segunda função de núcleo $K_2 = K_1 + K$, onde K representa a função de núcleo com a não-linearidade que se deseja adicionar à primeira função.

Com relação ao algoritmo multi-núcleo MKPLS, ao examinarmos os dois algoritmos de desconto, percebemos que as equações são análogas às usadas no LPLS para o cálculo residual no treinamento e predição. De fato, o MKPLS é semelhante ao algoritmo de apenas um núcleo, sendo que a principal diferença está no desconto necessário para a troca das funções núcleo durante o treinamento e predição. Um algoritmo de alto-nível é apresentado na figura 5.12.

Abordagem Multi-Kernel para regressão PLS

1 - Aplicar a regressão PLS de um núcleo em K_1
2 - Descontar da segunda matriz núcleo K_2
o modelo obtido no passo anterior
3 - Aplicar a regressão PLS de um núcleo
à matriz descontada K_2

Figura 5.12: Principais passos do MKPLS.

5.4.4 Resultados experimentais

Por simplicidade, para avaliar o impacto da estratégia MKPLS, fizemos sua aplicação à regressão PLS1. Para isso, utilizamos 5 conjuntos de dados do conjunto NIR: *Wheat, Meat, Combustible, Light gas oil* e *Corn*. A curva PRESS gerada pelo MKPLS foi comparada com a do PLS e LPLS, sendo que duas características foram observadas durante o experimento:

1. complexidade do modelo;

2. e, qualidade da predição.

A complexidade do modelo se refere ao número de fatores necessários para conseguir um erro de predição relativamente pequeno. Assim, comparamos as curvas PRESS para os primeiros fatores. Neste experimento, os 10 primeiros fatores foram considerados para a maioria dos conjuntos.

A qualidade de predição é dada pelo PRESS mínimo considerando todos os fatores. Para cada região mencionada, o mínimo de cada curva é comparado. Além disto, a porcentagem de vezes que o MKPLS obteve desempenho melhor ou igual aos outros modelos é calculada, já que os conjuntos foram amostrados 20 vezes.

Para cada conjunto, os seguintes parâmetros são convenientemente ajustados:

1. a primeira função de núcleo K_1 resultando na matriz K_1;
2. o número de fatores f_1 calculados com K_1;
3. a segunda função de núcleo K_2 com a matriz correspondente K_2.

Para todos os conjuntos, a função de núcleo identidade $K_1(\mathsf{u}, \mathsf{v}) = \mathsf{u}^\top \mathsf{v}$ foi usada, resultando na matriz $\mathsf{K}_1 = \mathsf{X}\mathsf{X}^\top$. Núcleos polinomiais ou gaussianos foram usados para K_2 em todos os experimentos. Independente do núcleo usado, parâmetros que mostraram bom desempenho em experimentos com o LPLS foram usados com o MKPLS.

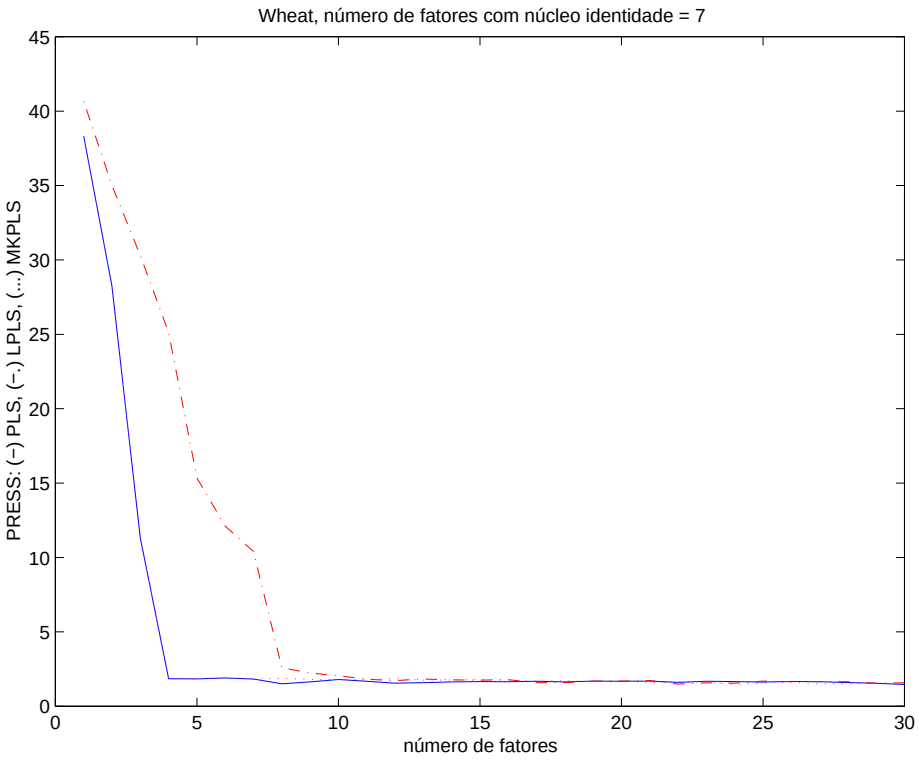

Figura 5.13: Curva PRESS de PLS, LPLS e MKPLS para o conjunto Wheat.

Para ilustrar o desempenho do MKPLS, as curvas PRESS para os três modelos são mostradas para alguns exemplos. Nas figuras 5.13 e 5.14

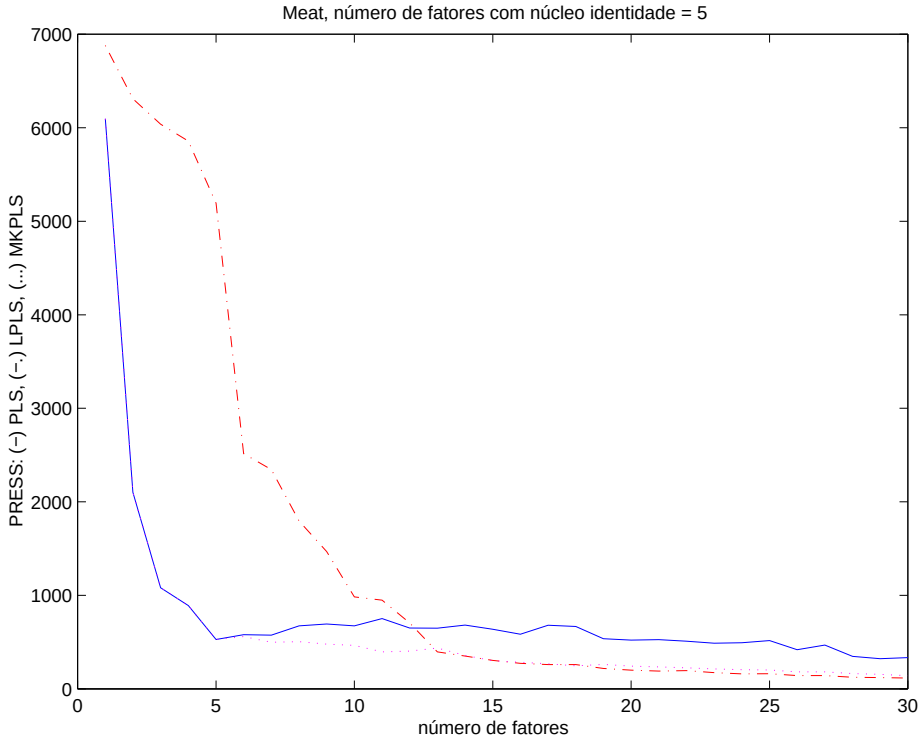

Figura 5.14: Curva PRESS de PLS, LPLS e MKPLS para o conjunto Meat.

constatamos como o MKPLS se beneficia das modelagens PLS e LPLS. O bom desempenho do modelo PLS para os primeiros fatores é aproveitado, enquanto a alta qualidade preditiva do modelo não-linear para os demais fatores, é mantida. A tabela 5.3 reporta o desempenho do MKPLS para todos os conjuntos de dados. Para os conjuntos *Wheat, Meat* e *Combustible* adotamos os 10 primeiros fatores. Já para os conjuntos *Light gas oil* e *Corn*, adotamos os 15 primeiros fatores. A tabela 5.4 mostra o ganho obtido com o MKPLS sobre o PLS e LPLS quando usados os 30 primeiros fatores. Observe que com mais de 30 fatores, não há nenhum acréscimo do poder preditivo dos modelos em exame, quando aplicados a qualquer um dos 5 conjuntos. Em ambas as tabelas, *MKPLS sobre PLS* indica o ganho obtido com o PRESS mínimo do MKPLS, quando comparado com o do PLS para o número de fatores indicados, ou seja,

$$100 \cdot \left(1 - \frac{min(PRESS_{MKPLS})}{min(PRESS_{PLS})}\right)$$

O mesmo se aplica a *MKPLS sobre LPLS*.

A seguir, comentamos detalhadamente os resultados obtidos, juntamente com os correspondentes parâmetros usados, para cada um dos 5 conjuntos de teste.

Tabela 5.3: Comparação do PRESS do MKPLS com PLS e LPLS para os primeiros fatores.

Conjunto	MKPLS sobre PLS			
	média	desvio	% empate	% vitória
Wheat	1.32	4.54	35.0	45.0
Meat	14.04	14.51	0.0	90.0
Combustible	3.09	11.86	45.0	45.0
Light gas oil	-0.32	2.34	75.0	20.0
Corn	1.91	3.01	45.0	55.0

Conjunto	MKPLS sobre LPLS			
	média	desvio	% empate	% vitória
Wheat	23.81	24.48	0.0	95.0
Meat	58.47	6.84	0.0	100.0
Combustible	45.60	40.14	0.0	95.0
Light gas oil	15.94	18.93	0.0	85.0
Corn	32.90	15.10	0.0	100.0

Tabela 5.4: Comparação do PRESS do MKPLS com PLS e LPLS para todos os fatores.

Conjunto	MKPLS sobre PLS			
	média	desvio	% empate	% vitória
Wheat	13.07	12.86	15.0	85.0
Meat	41.47	19.82	0.0	95.0
Combustible	29.36	24.76	25.0	75.0
Light gas oil	0.60	1.95	65.0	30.0
Corn	1.24	11.00	15.0	55.0

Conjunto	MKPLS sobre LPLS			
	média	desvio	% empate	% vitória
Wheat	6.52	11.89	0.0	80.0
Meat	-4.65	11.70	0.0	45.0
Combustible	48.82	47.36	0.0	90.0
Corn	18.81	16.61	0.0	90.0
Light gas oil	11.22	21.92	0.0	80.0

Wheat

O núcleo identidade foi usado para modelar os 7 primeiros fatores, e para K_2 foi usada a função de núcleo polinomial $(\mathsf{x}_i^\top \mathsf{x}_j + 1)^2$. A complexidade do modelo foi avaliada para os 10 primeiros fatores.

Como mostram as tabelas, e ilustrado também pela figura 5.13, o uso

do modelo híbrido permite melhor desempenho com poucos fatores, não somente sobre o LPLS como já era esperado, mas também sobre o PLS: em 80% das rodadas o MKPLS no mínimo igualou ao PLS, sendo melhor em 46,5% contra 20% do PLS.

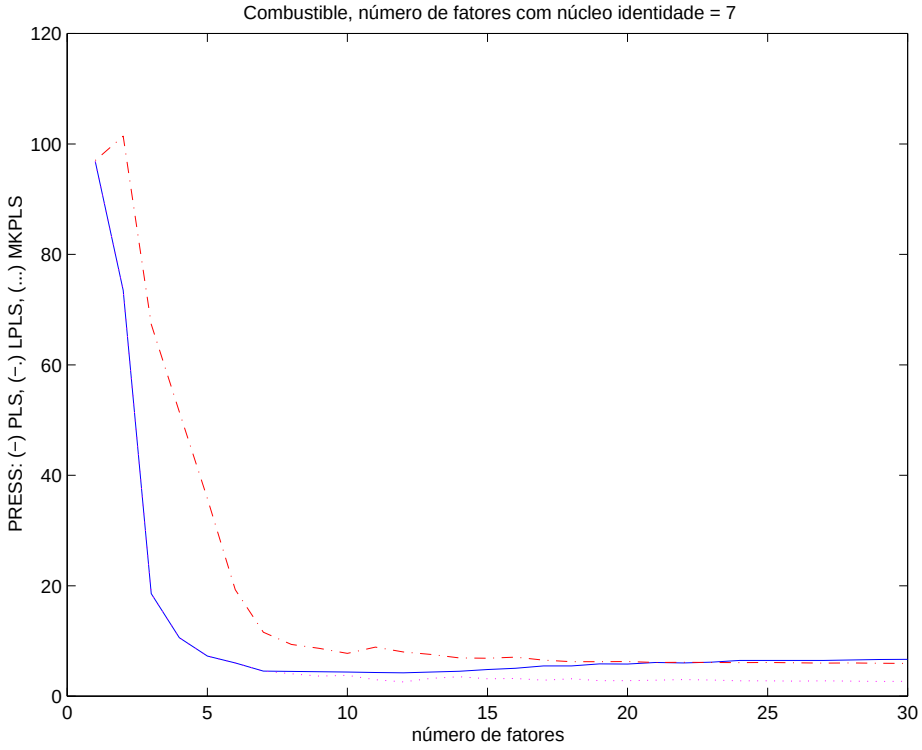

Figura 5.15: Curva PRESS de PLS, LPLS e MKPLS para o conjunto Combustible.

Meat

O núcleo identidade foi usado para modelar os 5 primeiros fatores, e para K_2 foi usada a função de núcleo polinomial $(\mathsf{x}_i^\top \mathsf{x}_j + 1)^2$.

O modelo conseguido com o MKPLS possui complexidade baixa, o uso do mapeamento identidade para modelar os 5 primeiros fatores não somente eliminou a modelagem pobre do LPLS, mas forneceu um melhor modelo do que o PLS. Em 90% das rodadas o MKPLS forneceu melhor PRESS para os 10 primeiros fatores. Considerando todos os fatores, tanto o MKPLS como o LPLS fornecem resultados nitidamente melhores do que o PLS. A figura 5.14 ilustra este resultado. Entre os dois modelos não-lineares, não há um vencedor, mas o desempenho conseguido com poucos fatores faz do MKPLS um melhor candidato.

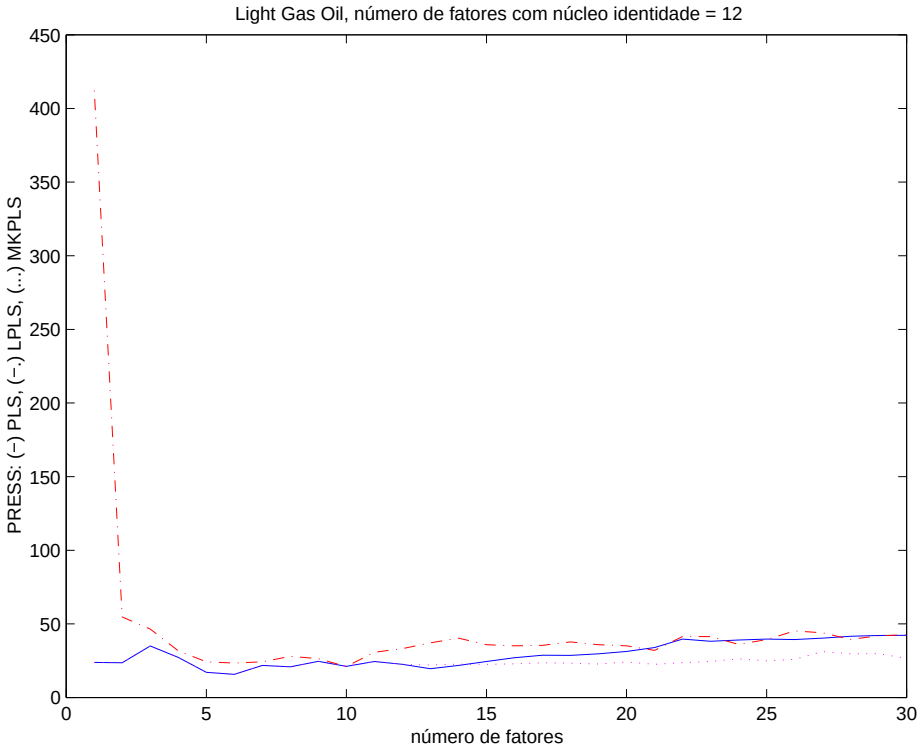

Figura 5.16: Curva PRESS de PLS, LPLS e MKPLS para o conjunto Light gas oil.

Combustible

Foi usada a função de núcleo polinomial $(\mathsf{x}_i^\top \mathsf{x}_j + 1)^2$ para a matriz K_2, e 7 fatores foram modelados como o núcleo identidade.

Novamente, a complexidade do modelo MKPLS é baixa. Avaliando as curvas para os 10 primeiros fatores, é fácil ver que o MKPLS possui desempenho no mínimo igual ao do PLS, e nitidamente superior ao do LPLS. Analisando os modelos para todos os fatores, vemos que o MKPLS não somente é superior ao PLS, mas também do LPLS.

A motivação inicial no desenvolvimento do MKPLS era usufruir de ambas as modelagens PLS e LPLS. Porém, os resultados com o conjunto *Combustible* mostram que o uso de um modelo linear para os primeiros fatores agrega ao MKPLS informação que melhora seu poder preditivo nos demais f_2 fatores não-lineares, sobrepondo-se ao LPLS.

Gas light oil

Para este conjunto, a avaliação de complexidade do modelo foi realizada nos 15 primeiros fatores, pois o próprio modelo PLS requer mais fatores para estabilizar a curva PRESS. Mesmo com 15 fatores para o MK-

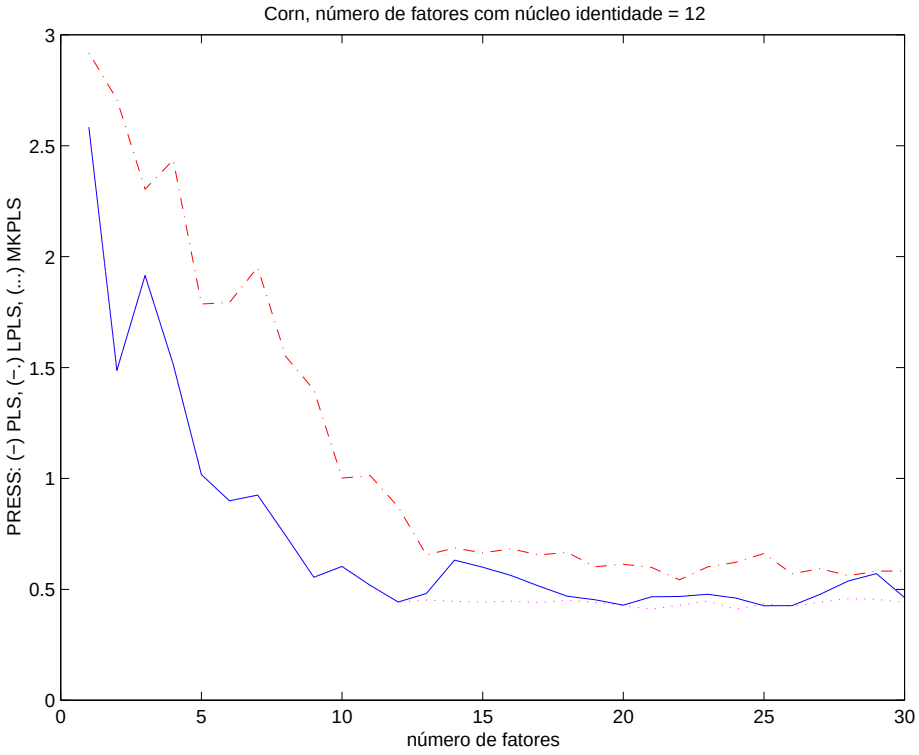

Figura 5.17: Curva PRESS de PLS, LPLS e MKPLS para o conjunto Corn.

PLS, ele ainda é mais compacto com relação ao número necessário de fatores para o modelo não linear LPLS.
A função de núcleo identidade foi usada para os 12 primeiros fatores, sendo que para os demais foi usado o núcleo gaussiano com desvio igual a 3.

Com relação ao PLS, não houve uma melhora expressiva, como ilustrado na figura 5.16. Considerando todos os fatores, os modelos MKPLS e PLS empataram 65% da vezes, ainda que o MKPLS se mostrou superior em 30%. A modelagem puramente não-linear não é adequada para este conjunto, porém a abordagem MKPLS oferece resultados no mínimo iguais ao PLS, sendo sua escolha apropriada.

Corn

Neste conjunto, a complexidade dos modelos foi avaliada nos 15 primeiros fatores pelos mesmos motivos apresentados para o conjunto anterior. O núcleo identidade foi usado nos 12 primeiros fatores. Para os demais a função de núcleo híbrida $K_2 = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^2 + e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{6^2}}$ foi usada.

Novamente, o modelo puramente não-linear LPLS tem dificuldades em modelar a variável dependente (figura 5.17), mas o uso do MKPLS fornece

um predição na maioria das vezes melhor do que a conseguida com o PLS, mostrando que a abordagem multi-núcleos é vantajosa.

5.4.5 Comentários

Com relação aos resultados do LPLS, mesmo que os parâmetros (função de núcleo e ajustes) não sejam ótimos, a comparação realizada com o MKPLS usou a mesma configuração. Se houver ajustes que levem a uma melhor predição com o LPLS, o modelo MKPLS também é beneficiado.

Com relação ao desvio, observamos grande variabilidade nas amostragens. Por este motivo foram reportados os casos vencedores e de empate. Na maioria das vezes, a existência de alguma rodada geradora de ganho muito alto, aumenta consideravelmente o desvio, apesar de pouco afetar a média. Por outro lado, os conjuntos usados contém poucas amostras, 215 no máximo, por serem oriundos da área de quimiometria. A aplicação dos modelos em conjuntos maiores, com população superior a 1000, deve fornecer resultados mais estáveis.