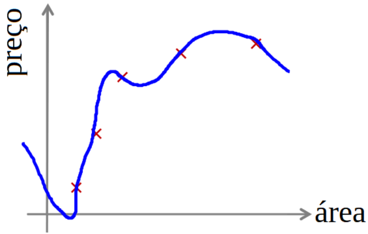
Aula 13 - Método do Kernel

João Florindo

Instituto de Matemática, Estatística e Computação Científica
Universidade Estadual de Campinas - Brasil
florindo@unicamp.br

Outline



- Vimos a regressão linear polinomial, em que, por exemplo, $y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$.

- Definimos a função vetorial $\phi : \mathbb{R} \to \mathbb{R}^4$ e o vetor $\theta \in \mathbb{R}^4$:

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^4 \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \tag{1}$$

  e assim

$$y = \theta^T \phi(x)$$

  e y é vista então como uma função linear sobre $\phi(x)$.

- Para distinguir, x é chamado de **atributo** e $\phi(x)$ de **feature** (**característica**) ou ainda **feature map**.

- Definimos a função vetorial $\phi : \mathbb{R} \to \mathbb{R}^4$ e o vetor $\theta \in \mathbb{R}^4$:

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^4 \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \tag{1}$$

  e assim

$$y = \theta^T \phi(x)$$

  e y é vista então como uma função linear sobre $\phi(x)$.

- Para distinguir, x é chamado de **atributo** e $\phi(x)$ de **feature** (**característica**) ou ainda **feature map**.

Mínimos Quadrados

- Original:

$$\begin{aligned} \theta &:= \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})x^{(i)} \\ &:= \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})x^{(i)}. \end{aligned} \quad (2)$$

- Seja agora $\phi : \mathbb{R}^n \to \mathbb{R}^p$, tal que em (1) temos $n = 1$ e $p = 4$.
- Então (2) se torna

$$\theta := \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)}).$$

Mínimos Quadrados

- Original:

$$\begin{aligned} \theta &:= \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})x^{(i)} \\ &:= \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})x^{(i)}. \end{aligned} \tag{2}$$

- Seja agora $\phi : \mathbb{R}^n \to \mathbb{R}^p$, tal que em (1) temos $n = 1$ e $p = 4$.

- Então (2) se torna

$$\theta := \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)}).$$

Mínimos Quadrados

- Original:

$$\begin{aligned}\theta &:= \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})x^{(i)} \\ &:= \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})x^{(i)}.\end{aligned} \tag{2}$$

- Seja agora $\phi : \mathbb{R}^n \to \mathbb{R}^p$, tal que em (1) temos $n = 1$ e $p = 4$.

- Então (2) se torna

$$\theta := \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)}).$$

Problema

- Suponha $x \in \mathbb{R}^n$ e ϕ um vetor com monômios de x de grau ≤ 3:

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_2 x_1 \\ \vdots \\ x_1^3 \\ x_1^2 x_2 \\ \vdots \end{bmatrix} \tag{3}$$

- $\phi(x)$ tem comprimento da ordem de n^3 (assumindo que a ordem dos fatores importa).

Problema

- EX.: Se $n = 1000$, $\phi(x)$ tem dimensão 10^9.
- 10^6 vezes maior que x original.
- Computacionalmente proibitivo!
- Seria possível reduzir este custo? SIM!

Problema

- EX.: Se $n = 1000$, $\phi(x)$ tem dimensão 10^9.
- 10^6 vezes maior que x original.
- Computacionalmente proibitivo!
- Seria possível reduzir este custo? SIM!

Problema

- EX.: Se $n = 1000$, $\phi(x)$ tem dimensão 10^9.
- 10^6 vezes maior que x original.
- Computacionalmente proibitivo!
- Seria possível reduzir este custo? SIM!

Problema

- EX.: Se $n = 1000$, $\phi(x)$ tem dimensão 10^9.
- 10^6 vezes maior que x original.
- Computacionalmente proibitivo!
- Seria possível reduzir este custo? SIM!

Truque do Kernel

- Podemos escrever
$$\theta = \sum_{i=1}^{m} \beta_i \phi(x^{(i)})$$
para algum $\beta_1, \cdots, \beta_m \in \mathbb{R}$.

Demonstração (por indução)

Assumimos a inicialização $\theta = 0$. No caso:

$$\theta = 0 = \sum_{i=1}^{m} 0 \cdot \phi(x^{(i)}).$$

Suponha que em dado passo temos $\theta = \sum_{i=1}^{m} \beta_i \phi(x^{(i)})$. Temos que mostrar que, no próximo passo, θ ainda é uma combinação linear de $\phi(x^{(i)})$.

Truque do Kernel

- Podemos escrever
$$\theta = \sum_{i=1}^{m} \beta_i \phi(x^{(i)})$$
para algum $\beta_1, \cdots, \beta_m \in \mathbb{R}$.

Demonstração (por indução)

Assumimos a inicialização $\theta = 0$. No caso:

$$\theta = 0 = \sum_{i=1}^{m} 0 \cdot \phi(x^{(i)}).$$

Suponha que em dado passo temos $\theta = \sum_{i=1}^{m} \beta_i \phi(x^{(i)})$. Temos que mostrar que, no próximo passo, θ ainda é uma combinação linear de $\phi(x^{(i)})$.

Truque do Kernel

Demonstração (continuação)

Próximo passo:

$$
\begin{aligned}
\theta &:= \theta + \alpha \sum_{i=1}^{m} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)}) \\
&= \sum_{i=1}^{m} \beta_i \phi(x^{(i)}) + \alpha \sum_{i=1}^{m} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)}) \\
&= \sum_{i=1}^{m} \underbrace{(\beta_i + \alpha(y^{(i)} - \theta^T \phi(x^{(i)})))}_{\text{novo } \beta_i} \phi(x^{(i)})
\end{aligned}
\tag{4}
$$

Truque do Kernel

- Ideia é representar $\theta \in \mathbb{R}^p$ por $\beta \in \mathbb{R}^n$ $(n \ll p)$.

- De (4) temos a atualização:

$$\beta_i := \beta_i + \alpha(y^{(i)} - \theta^T \phi(x^{(i)})).$$

- Substituindo θ por $\sum_{j=1}^{m} \beta_j \phi(x^{(j)})$:

$$\begin{aligned}\beta_i &:= \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{m} \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right) \\ &:= \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{m} \beta_j \langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle \right).\end{aligned}$$

Truque do Kernel

- Ideia é representar $\theta \in \mathbb{R}^p$ por $\beta \in \mathbb{R}^n$ ($n \ll p$).

- De (4) temos a atualização:

$$\beta_i := \beta_i + \alpha(y^{(i)} - \theta^T \phi(x^{(i)})).$$

- Substituindo θ por $\sum_{j=1}^m \beta_j \phi(x^{(j)})$:

$$\begin{aligned}\beta_i &:= \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^m \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right) \\ &:= \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^m \beta_j \langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle \right).\end{aligned}$$

Truque do Kernel

- Ideia é representar $\theta \in \mathbb{R}^p$ por $\beta \in \mathbb{R}^n$ ($n \ll p$).
- De (4) temos a atualização:

$$\beta_i := \beta_i + \alpha(y^{(i)} - \theta^T \phi(x^{(i)})).$$

- Substituindo θ por $\sum_{j=1}^{m} \beta_j \phi(x^{(j)})$:

$$\begin{aligned}\beta_i &:= \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{m} \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right) \\ &:= \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{m} \beta_j \langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle \right).\end{aligned}$$

Truque do Kernel

- Mas ainda precisamos calcular $\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$? Sim, MAS ...

1. $\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$ podem ser pré-calculados antes das iterações começarem.
2. Para o *feature map* em (3) (e muitos outros), o produto interno se calcula de forma eficiente (sem precisar de $\phi(x^{(i)})$):

$$\begin{aligned} \langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle &= 1 + \sum_{i=1}^{n} x_i z_i + \sum_{i,j \in \{1,\cdots,n\}} x_i x_j z_i z_j + \sum_{i,j,k \in \{1,\cdots,n\}} x_i x_j x_k z_i z_j z_k \\ &= 1 + \sum_{i=1}^{n} x_i z_i + \left(\sum_{i=1}^{n} x_i z_i \right)^2 + \left(\sum_{i=1}^{n} x_i z_i \right)^3 \\ &= 1 + \langle x, z \rangle + \langle x, z \rangle^2 + \langle x, z \rangle^3. \end{aligned} \tag{5}$$

Truque do Kernel

- Mas ainda precisamos calcular $\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$? Sim, MAS ...

1. $\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$ podem ser pré-calculados antes das iterações começarem.
2. Para o *feature map* em (3) (e muitos outros), o produto interno se calcula de forma eficiente (sem precisar de $\phi(x^{(i)})$):

$$\begin{aligned} \langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle &= 1 + \sum_{i=1}^{n} x_i z_i + \sum_{i,j \in \{1,\cdots,n\}} x_i x_j z_i z_j + \sum_{i,j,k \in \{1,\cdots,n\}} x_i x_j x_k z_i z_j z_k \\ &= 1 + \sum_{i=1}^{n} x_i z_i + \left(\sum_{i=1}^{n} x_i z_i \right)^2 + \left(\sum_{i=1}^{n} x_i z_i \right)^3 \\ &= 1 + \langle x, z \rangle + \langle x, z \rangle^2 + \langle x, z \rangle^3. \end{aligned} \tag{5}$$

Truque do Kernel

Definição

O **Kernel** associado a ϕ é a função $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ definida por:

$$K(x, z) \triangleq \langle \phi(x), \phi(z) \rangle.$$

Truque do Kernel

Algoritmo:

1. Para todo $i, j \in \{1, \cdots, m\}$ calcular $K(x^{(i)}, x^{(j)})$ usando (5) e fazer $\beta := 0$.
2. Iterar:
$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{m} \beta_j K(x^{(i)}, x^{(j)}) \right).$$
Vetorialmente, definimos a matriz de *kernel* K tal que $K_{ij} = K(x^{(i)}, x^{(j)})$ e
$$\beta := \beta + \alpha(y - K\beta).$$
3. Por fim, para calcular uma predição $\theta^T \phi(x)$, usamos
$$\theta^T \phi(x) = \sum_{i=1}^{m} \beta_i \phi(x^{(i)})^T \phi(x) = \sum_{i=1}^{m} \beta_i K(x^{(i)}, x).$$

Truque do Kernel

Algoritmo:

1. Para todo $i, j \in \{1, \cdots, m\}$ calcular $K(x^{(i)}, x^{(j)})$ usando (5) e fazer $\beta := 0$.
2. Iterar:

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{m} \beta_j K(x^{(i)}, x^{(j)}) \right).$$

Vetorialmente, definimos a matriz de *kernel* K tal que $K_{ij} = K(x^{(i)}, x^{(j)})$ e

$$\beta := \beta + \alpha(y - K\beta).$$

3. Por fim, para calcular uma predição $\theta^T \phi(x)$, usamos

$$\theta^T \phi(x) = \sum_{i=1}^{m} \beta_i \phi(x^{(i)})^T \phi(x) = \sum_{i=1}^{m} \beta_i K(x^{(i)}, x).$$

Truque do Kernel

Algoritmo:

1. Para todo $i, j \in \{1, \cdots, m\}$ calcular $K(x^{(i)}, x^{(j)})$ usando (5) e fazer $\beta := 0$.
2. Iterar:
$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{m} \beta_j K(x^{(i)}, x^{(j)}) \right).$$
Vetorialmente, definimos a matriz de *kernel* K tal que $K_{ij} = K(x^{(i)}, x^{(j)})$ e
$$\beta := \beta + \alpha(y - K\beta).$$
3. Por fim, para calcular uma predição $\theta^T \phi(x)$, usamos
$$\theta^T \phi(x) = \sum_{i=1}^{m} \beta_i \phi(x^{(i)})^T \phi(x) = \sum_{i=1}^{m} \beta_i K(x^{(i)}, x).$$

Propriedades

- Que tipo de função $K(\cdot, \cdot)$ pode ser um *kernel*, i.e., corresponde a algum *feature map* ϕ?

Teorema (Mercer)

Seja $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Então K é um *kernel* (de Mercer) válido **se e somente se** para quaisquer $\{x^{(1)}, \cdots, x^{(n)}\}$ $(n < \infty)$, a matriz de *kernel* correspondente é simétrica e positiva semi-definida.

Propriedades

- Que tipo de função $K(\cdot, \cdot)$ pode ser um *kernel*, i.e., corresponde a algum *feature map* ϕ?

Teorema (Mercer)

Seja $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Então K é um *kernel* (de Mercer) válido **se e somente se** para quaisquer $\{x^{(1)}, \cdots, x^{(n)}\}$ $(n < \infty)$, a matriz de *kernel* correspondente é simétrica e positiva semi-definida.

Demonstração da Condição Necessária

- Se K é um *kernel* válido, então $K_{ij} = K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)}) = \phi(x^{(j)})^T \phi(x^{(i)}) = K(x^{(j)}, x^{(i)}) = K_{ji}$, e portanto K é simétrica.

Demonstração da Condição Necessária

- Ademais, seja $\phi_k(x)$ a k-ésima coordenada do vetor $\phi(x)$. Para um vetor qualquer z temos então:

$$\begin{aligned}
z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\
&= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \\
&\geq 0,
\end{aligned}$$

em que o penúltimo passo usa $\sum_{i,j} a_i a_j = (\sum_i a_i)^2$ com $a_i = z_i \phi_k(x^{(i)})$. Como z é arbitrário, isso mostra que K é positiva semi-definida.

Outros Exemplos

- $$K(x,z) = (x^T z)^2$$
- Pode ser escrito como

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_j z_j\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \end{aligned}$$

Outros Exemplos

e cujo *feature map*, p.ex, para $n = 3$ é

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}.$$

Outros Exemplos

$$
\begin{aligned}
K(x,z) &= (x^T z + c)^2 \\
&= \sum_{i,j}^{n} (x_i x_j)(z_i z_j) + \sum_{j=1}^{n} (\sqrt{2c}x_i)(\sqrt{2c}z_i) + c^2,
\end{aligned}
$$

para o qual

$$
\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ \sqrt{2c}x_3 \\ c \end{bmatrix}.
$$

Kernel como Medida de Similaridade

- Intuitivamente, $K(x, z)$ assume um valor pequeno quando $\phi(x)$ e $\phi(z)$ estão distantes entre si (p.ex., quase ortogonais) e vice-versa.

- Assim, $K(x, z)$ mede o quão similares são $\phi(x)$ e $\phi(z)$ (ou x e z).

- EX.: **Kernel Gaussiano**:

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right).$$

Kernel como Medida de Similaridade

- Intuitivamente, $K(x, z)$ assume um valor pequeno quando $\phi(x)$ e $\phi(z)$ estão distantes entre si (p.ex., quase ortogonais) e vice-versa.
- Assim, $K(x, z)$ mede o quão similares são $\phi(x)$ e $\phi(z)$ (ou x e z).
- EX.: **Kernel Gaussiano**:

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right).$$

Kernel como Medida de Similaridade

- Intuitivamente, $K(x,z)$ assume um valor pequeno quando $\phi(x)$ e $\phi(z)$ estão distantes entre si (p.ex., quase ortogonais) e vice-versa.
- Assim, $K(x,z)$ mede o quão similares são $\phi(x)$ e $\phi(z)$ (ou x e z).
- EX.: **Kernel Gaussiano**:

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right).$$

Aplicabilidade

- Exemplificamos na regressão linear.
- Mas a ideia de *kernel* pode ser aplicada a qualquer algoritmo que possa ser escrito em termos do produto interno entre vetores de atributos.
- EX.: Regressão logística, SVM, etc.

Aplicabilidade

- Exemplificamos na regressão linear.
- Mas a ideia de *kernel* pode ser aplicada a qualquer algoritmo que possa ser escrito em termos do produto interno entre vetores de atributos.
- EX.: Regressão logística, SVM, etc.

Aplicabilidade

- Exemplificamos na regressão linear.
- Mas a ideia de *kernel* pode ser aplicada a qualquer algoritmo que possa ser escrito em termos do produto interno entre vetores de atributos.
- EX.: Regressão logística, SVM, etc.