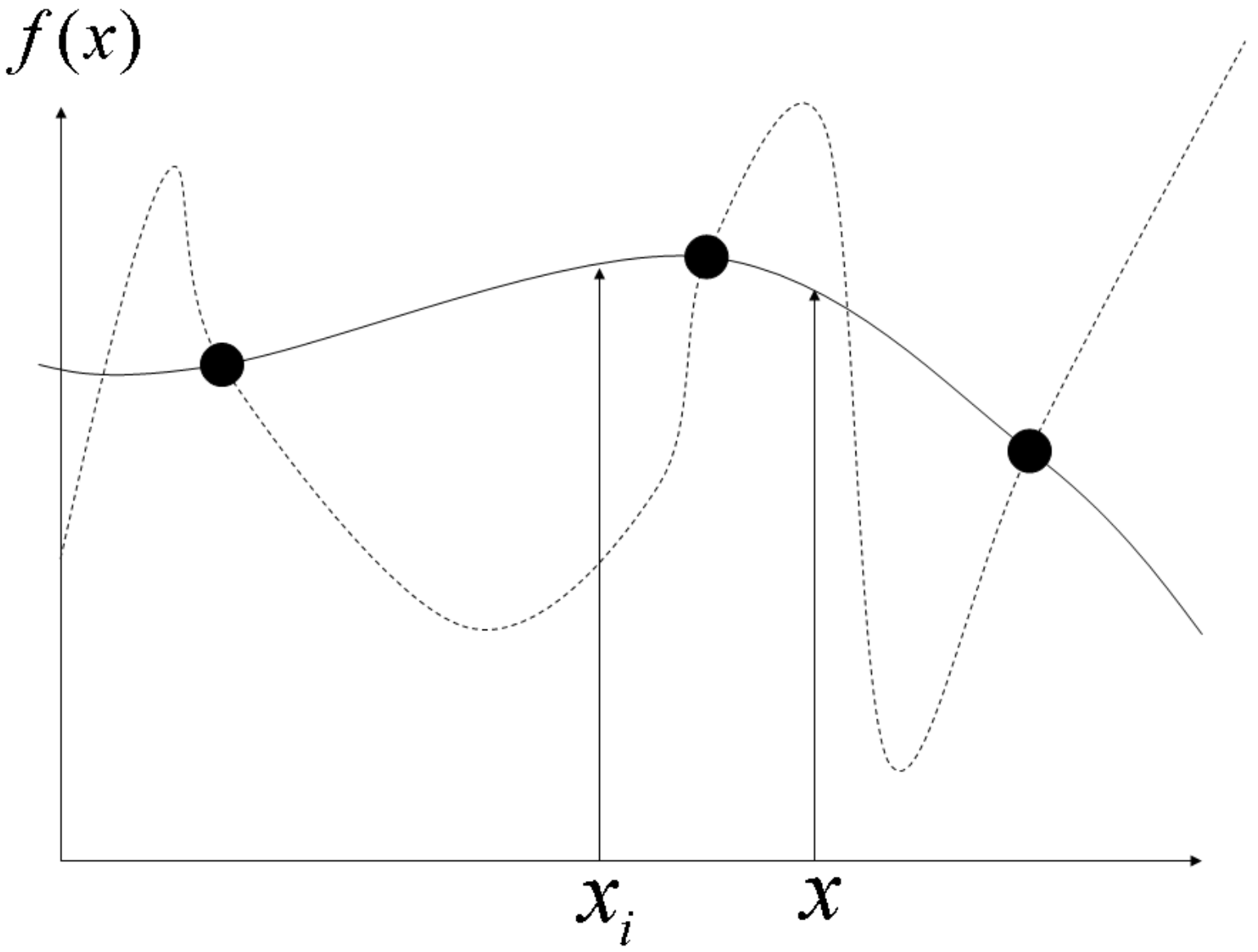
Kerne

Volker Tresp

Glattheitsannahme

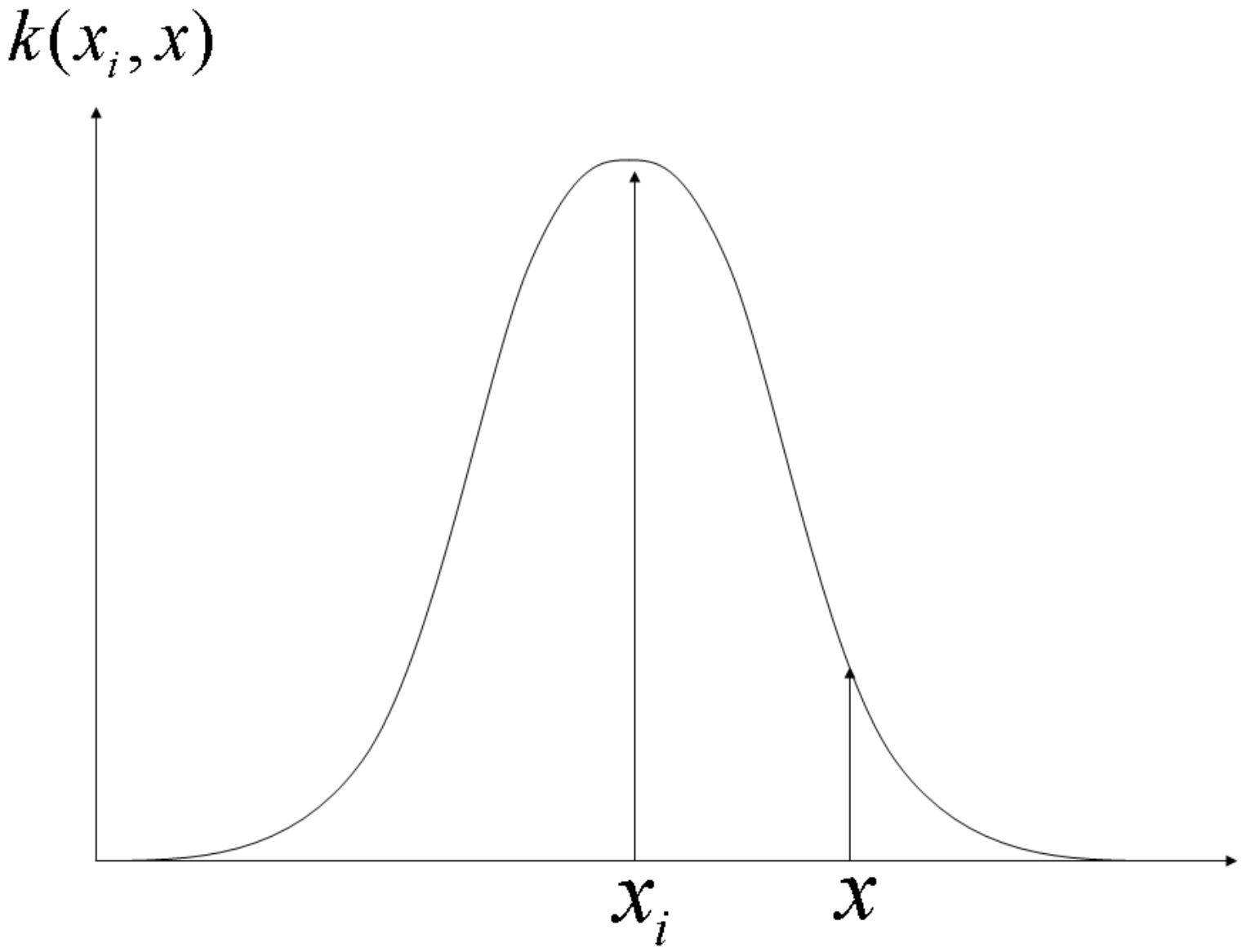
- Bisher haben wir Vorwissen einbringen können, indem wir geeignete Basisfunktionen definiert haben
- Alternativ mag es sinnvoll sein, glatte Funktionen zu bevorzugen: Funktionswerte an benachbarte Eingangswerten sollen ähnlich sein
- In der Abbildung mag es Sinn machen, anzunehmen, dass die Funktionswerte bei $\mathbf{x}_i$ und $\mathbf{x}$ ähnlich sind (Glattheitsannahme)
- Daher würde man die kontinuierliche Funktion der gestrichelten vorziehen


$f(x)$
x_i
x

Einführung Kerne

- Man kann diese Glattheitsannahme über Kernfunktionen implementieren
- Eine Kernfunktion $k(\mathbf{x}_i, \mathbf{x})$ gibt an, wie benachbarte Funktionswerte $f(x)$ sich verhalten, wenn $f(\mathbf{x}_i)$ gegeben ist
- Die Abbildung zeigt als Beispiel einen Gauß-schen Kerns
- Es stellt sich heraus, dass es eine enge Beziehung zwischen Modellen mit festen Basisfunktionen und Kernmodellen gibt:

$$k(\mathbf{x}_i, \mathbf{x}) = \sum_{j=1}^{M_\phi} \phi_j(\mathbf{x}_i)\phi_j(\mathbf{x})$$


$k(x_i, x)$
x_i
x

Umformungen der Kostenfunktion

- Wir beginnen mit der PLS Kostenfunktion für Modelle mit Basisfunktionen
- Regularisierte Kostenfunktion

$$\text{cost}^{pen}(\mathbf{w}) = \sum_{i=1}^{N}(y_i - \sum_j w_j \phi_j(x_i))^2 + \lambda \sum_{i=0}^{M} w_i^2$$

$$= (\mathbf{y} - \mathbf{\Phi w})^T(\mathbf{y} - \mathbf{\Phi w}) + \lambda \mathbf{w}^T\mathbf{w}$$

wobei Φ die design matrix ist mit $(\Phi)_{i,j} = \phi_j(\mathbf{x}_i)$.

- Wir setzen die erste Ableitung nach den Parametern gleich Null

$$\frac{\partial \text{cost}^{pen}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi w}) + 2\lambda \mathbf{w} = 0$$

Daraus folgt, dass man schreiben kann

$$\mathbf{w}_{pen} = \frac{1}{\lambda}\mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi w}_{pen})$$

- Dies ist nicht die Lösung (w erscheint auf beiden Seiten der Gleichung). Aber wir wissen nun, dass man die Lösung als lineare Kombination der Eingangsvektoren schreiben kann

$$\mathbf{w}_{pen} = \mathbf{\Phi}^T \mathbf{v} = \sum_{i=1}^{N} v_i \phi(\mathbf{x_i})$$

- Beachte, dass die Summe über die N Datenpunkte geht!

Eine neue Kostenfunktion

- Dies ist nur eine implizite Definition der Lösung; wir können nun jedoch dies als Nebenbedingung verwenden und in die Kostenfunktion einsetzten und erhalten

$$\text{cost}^{pen}(\mathbf{v}) = (\mathbf{y} - \mathbf{\Phi}\mathbf{\Phi}^T\mathbf{v})^T(\mathbf{y} - \mathbf{\Phi}\mathbf{\Phi}^T\mathbf{v}) + \lambda\mathbf{v}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{v}$$

$$= (\mathbf{y} - K\mathbf{v})^T(\mathbf{y} - K\mathbf{v}) + \lambda\mathbf{v}^T K\mathbf{v}$$

  wobei K eine $N \times N$ Matrix ist mit Elementen

$$k_{i,j} = \phi(\mathbf{x_i})^T\phi(\mathbf{x_j}) = \sum_{j=1}^{M_\phi} \phi_k(x_i)\phi_k(x_j)$$

  und mit

$$\mathbf{v} = (v_1, \ldots, v_N)^T$$

- Ein wichtiges Ergebnis: **Das Optimierungsproblem können wir so schreiben, dass nur die inneren Produkte der Basisvektoren auftauchen, aber nicht die Basisvektoren selber!**

Kern Gewichte

- Wir können nun die Kostenfunktion nach $\mathbf{v}$ ableiten (beachte, dass $K = K^T$)

$$\frac{\partial J_N^{pen}(\mathbf{v})}{\partial \mathbf{v}} = 2K(\mathbf{y} - K\mathbf{v}) + 2\lambda K\mathbf{v}$$

So dass

$$\mathbf{v}_{pen} = (K + \lambda I)^{-1}\mathbf{y}$$

Kern Vorhersage

- Eine Vorhersage lässt sich somit schreiben als

$$\widehat{f}(\mathbf{z}) = \phi(\mathbf{z})^T \mathbf{w} = \phi(\mathbf{z})^T \mathbf{\Phi}^T \mathbf{v}_{pen} = \sum_{i=1}^{N} v_i k(\mathbf{z}, \mathbf{x}_i)$$

  Mit

$$k(\mathbf{z}, \mathbf{x}_i) = \phi(\mathbf{z})^T \phi(\mathbf{x_i})$$

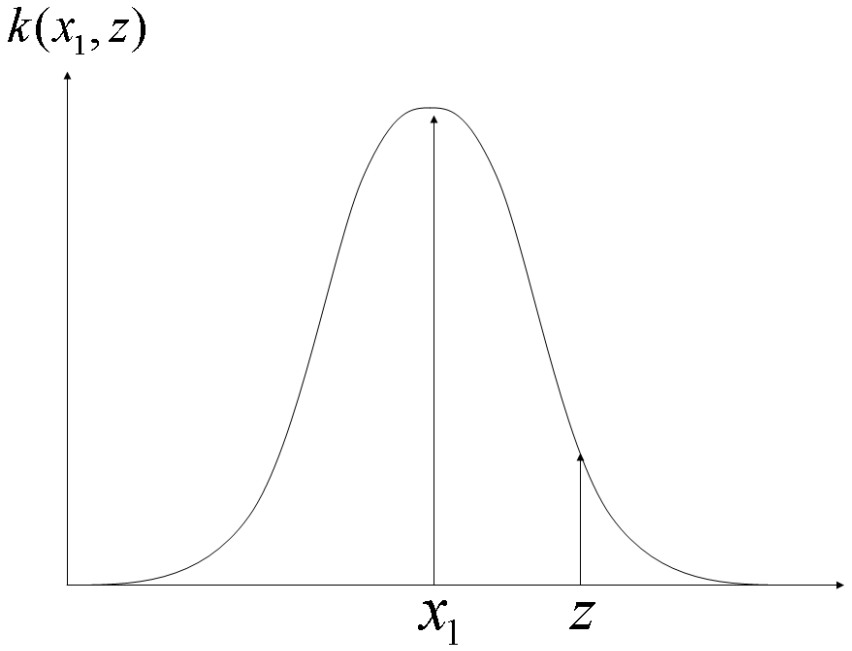
- Wieder ein wichtiges Resultat: auch die Vorhersage lässt sich so schreiben, dass nur innere Produkte verwendet werden; **die Lösung läßt sich als gewichtete Summe von N Kernfunktionen schreiben**.

Mit einem Trainingsdatenpunkt

- Mit nur einem Datenpunkt ergibt sich

$$f(\mathbf{z}) = v_1 k(\mathbf{z}, \mathbf{x}_1)$$

- Wie vorher diskutiert:



Bemerkungen und Interpretation des Kerns

- Dies ist nun schon interessanter, da es durchaus mehr Basisfunktionen als Eingangsdimensionen geben kann; insbesondere gilt das Resultat auch, wenn man mit **unendlich vielen Basisfunktionen** arbeitet
- Man kann sogar direkt mit den Kernfunktionen arbeiten, ohne sich besondere Gedanken über die Basisfunktionen zu machen, aus denen sie hergeleitet wurden
- Interpretation des Kerns
  - Als inneres Produkt $k(\mathbf{x}_i, \mathbf{z}) = \phi^T(\mathbf{x})\phi(\mathbf{z})$
  - Als Kovarianz: wie stark sind Funktionswerte an verschiedenen Eingangspunkten korreliert $k(\mathbf{x}_i, \mathbf{z}) = cov(f(\mathbf{x}_i), f(\mathbf{z}))$
- Wenn $N >> M$ ist die originale Formulierung im Merkmalsraum effizienter; wenn $M >> N$, ist die Kern-Version effizienter; genauer: im Merkmalsraum benötigt man $M^3 + M^2N$ Operationen und mit Kernen benötigt man $N^3 + N^2M$ Operationen. Wenn die Kerne a prior bekannt sind benötigt man N^3 Operationen

- In speziell strukturierten Problemen lassen sich Kerne berechnen, ohne explizit die NM^2 Operationen ausführen zu müssen. Beispiel: String Kerne. Dies ist eines der wichtigsten aktiven Forschungsaufgaben!

- Dennoch sind nicht alle Funktionen geeignete Kernfunktionen; dies stellt das folgende Theorem dar ...

Mercer Theorem

- Nach Vapnik: The nature of statistical learning theory
- *Mercer Theorem:* Um zu garantieren, dass die symmetrische Funktion $k(\mathbf{x}, \mathbf{z})$ aus L_2 eine Entwicklung der Art

$$k(\mathbf{x}, \mathbf{z}) = \sum_{h=1}^{\infty} \lambda_h \phi_h^T(\mathbf{x}) \phi_h(\mathbf{z})$$

besitzt, mit positiven Koeffizienten $\lambda_h > 0$, so ist es notwendig und ausreichend, dass

$$\int \int k(\mathbf{x}, \mathbf{z}) g(\mathbf{x}) g(\mathbf{z}) d\mathbf{x} d\mathbf{z} > 0$$

gültig ist für alle $g \neq 0$ für welche

$$\int g^2(\mathbf{x}) d\mathbf{x} < \infty$$

- Das Theorem sagt aus, dass für sogenannte positiv-definite Kerne, eine Zerlegung in Basisfunktionen möglich ist!

- Jede Kern-Matrix K ist dann ebenfalls positiv definit

Kern Design

- Linearer Kern

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

Hier sind sowohl Basisfunktionen als auch die entsprechenden Kerne lineare Funktionen

- Polynomialer Kern

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$$

Hier sind die entsprechenden Basisfunktionen alle geordneten Polynome des Grades d

- Polynomialer Kern (2)

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + R)^d$$

Hier sind die entsprechenden Basisfunktionen alle geordneten Polynome vom Grad d oder kleiner

- Gauß-Kern (RBF-Kern)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2s^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

Diese Kerne entsprechen *unendlich vielen* Gauß-förmigen Basisfunktionen

- Sigmoider Kern

$$k(\mathbf{x}_i, \mathbf{x}_j) = \text{sig}\left(\mathbf{x}_i^T \mathbf{x}_j\right)$$

Interessant im Zusammenhang mit Neuronalen Netze