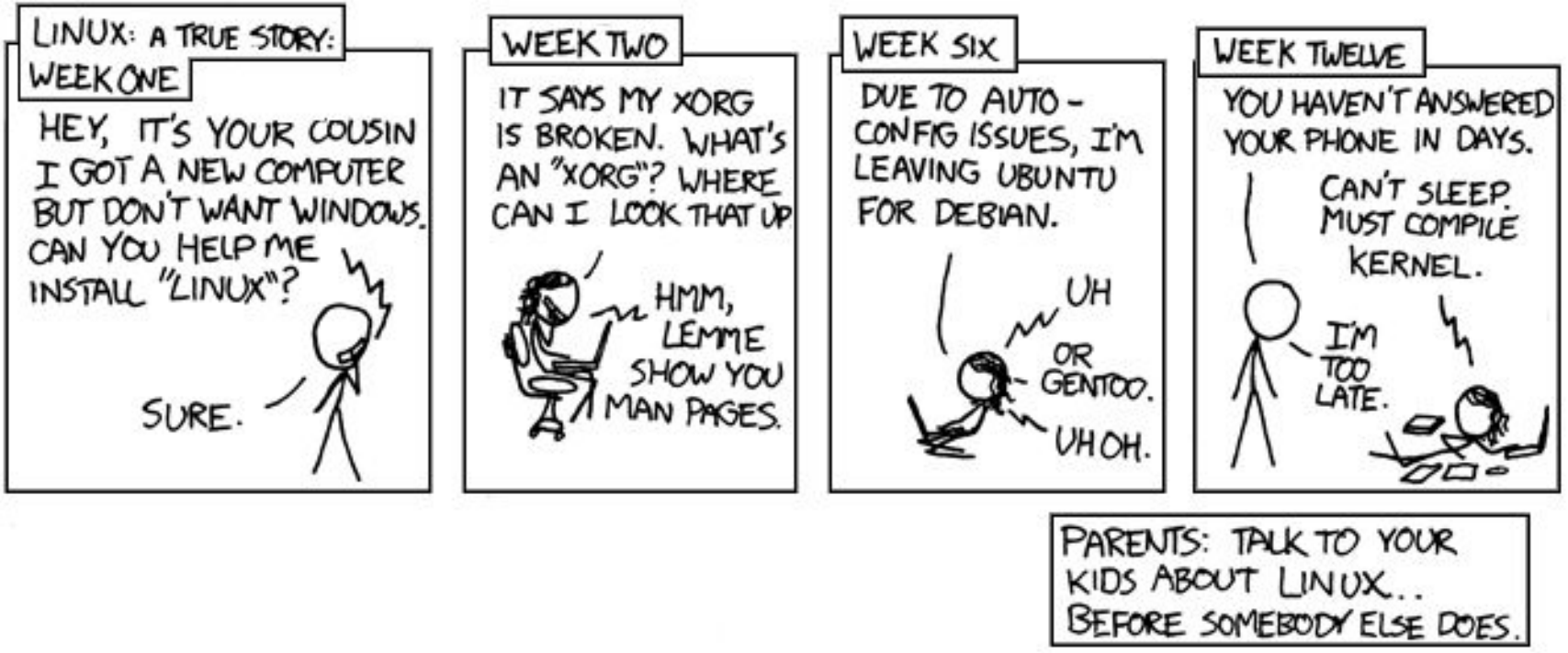
https://xkcd.com/456/

Not the same kind of kernels ...



Announcements

From the course reps – Taylor Qin and Yuchen Fang also see piazza @229

Education info session: FAQ on admin your degree

Topic: How to Admin Your Degree
Date: Friday 1 April 10am
Venue: Rm 1.36 Birch Building (35) or
https://anu.zoom.us/j/88641270633?pwd=ZU9paE1nUzJ0SU90c1krRnBJMWpGUT09

Facilitator: Paul Dowden, Manager, Student Services, Employability & Experience

About this event

There are a number of important elements CECS coursework students need to be aware of when managing their degree. The session will include:

1. Terminology (program/degree, major/minor; maximum/minimum etc.)
2. Building a degree:
    - Programs and Courses (P&C)
    - units of credit
    - Compulsory courses
    - Program List electives
    - 'Free' electives
    - Minors - Majors/Specialisations what are they and how and when to register
3. Program planning using P&C
4. What to do if it goes wrong!
    - Repeating a course
    - Reduced Study load
    - Program transfers incl. Credit & Exemption

After the break:

Week 7 Monday - Easter; Week 8 Monday - ANZAC day: no lecture or tutorial, join another online tutorial the same week.

Kernels

Basis functions recap

Non-parametric methods (Chap 2.5)

Dual representation (of linear models, Chap 6.1)

Constructing kernels (Chap 6.2)

Next time: maximum-margin classifiers (Chap 7.1)

Kernel

From Wikipedia, the free encyclopedia

Not to be confused with Colonel.

Kernel may refer to:

Look up *kernel* in Wiktionary, the free dictionary.

Contents [hide]

Computing [edit]

- Kernel (operating system), the central component of most operating systems
- Kernel (image processing), a matrix used for image convolution
- Compute kernel, in GPGPU programming
- Kernel method, in machine learning
- In numerical analysis, a subroutine that performs a common numerical operation
  - In particular, a routine that is executed in a vectorized loop; see, e.g., General-purpose computing on graphics processing units § Kernels
- Kernelization, a technique for designing efficient algorithms

Mathematics [edit]

Objects [edit]

- Kernel (algebra), a general concept that includes:
  - Kernel (linear algebra) or null space, a set of vectors mapped to the zero vector
  - Kernel (category theory), a generalization of the kernel of a homomorphism
  - Kernel (set theory), an equivalence relation: partition by image under a function
  - Difference kernel, a binary equalizer: the kernel of the difference of two functions

Functions [edit]

- Kernel (geometry), the set of points within a polygon from which the whole polygon boundary is visible
- Kernel (statistics), a weighting function used in kernel density estimation to estimate the probability density function of a random variable
- Integral kernel or **kernel function**, a function of two variables that defines an integral transform
- Heat kernel, the fundamental solution to the heat equation on a specified domain
- Convolution kernel
- Stochastic kernel, the transition function of a stochastic process
- Transition kernel, a generalization of a stochastic kernel
- Pricing kernel, the stochastic discount factor used in mathematical finance
- Positive-definite kernel, a generalization of a positive-definite matrix
- Kernel trick, in statistics
- Reproducing kernel Hilbert space

Natural sciences [edit]

- Seed, inside the nut of most plants, especially:

Recap: basis functions in linear models for classification

Linear w.r.t. Input x

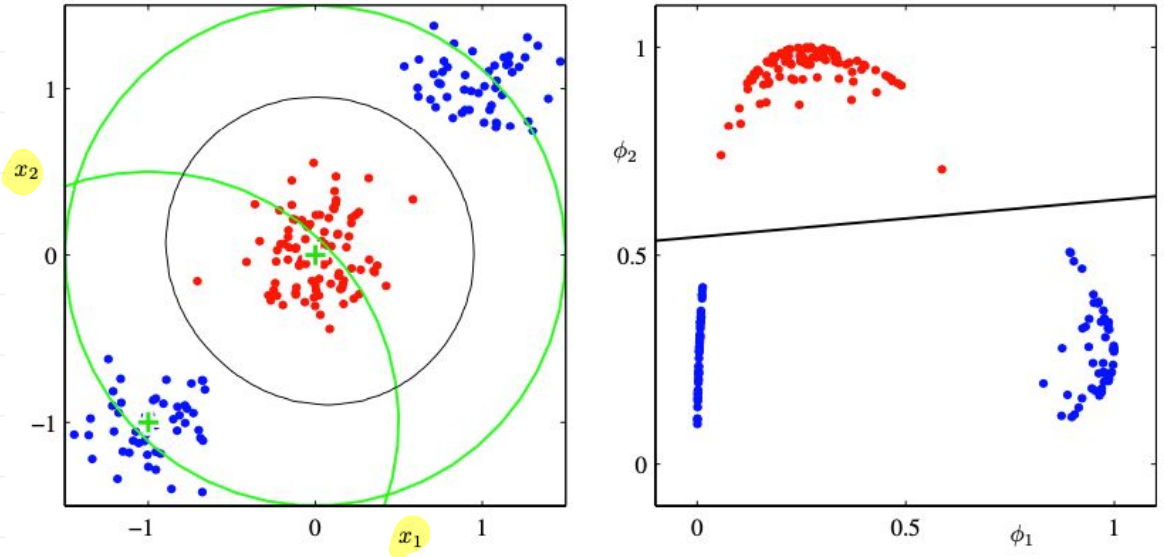
$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \tag{4.4}$$

Linear w.r.t. basis $\boldsymbol{\phi}$(x)

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\right) \tag{4.87}$$

$(x_1, x_2) \rightarrow (\phi_1(x_1, x_2), \phi_2(x_1, x_2))$

Fig 4.12



Recap: basis in linear models for regression

Linear w.r.t. input x

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

Linear w.r.t. basis $\boldsymbol{\phi}$(x)

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

fixed width

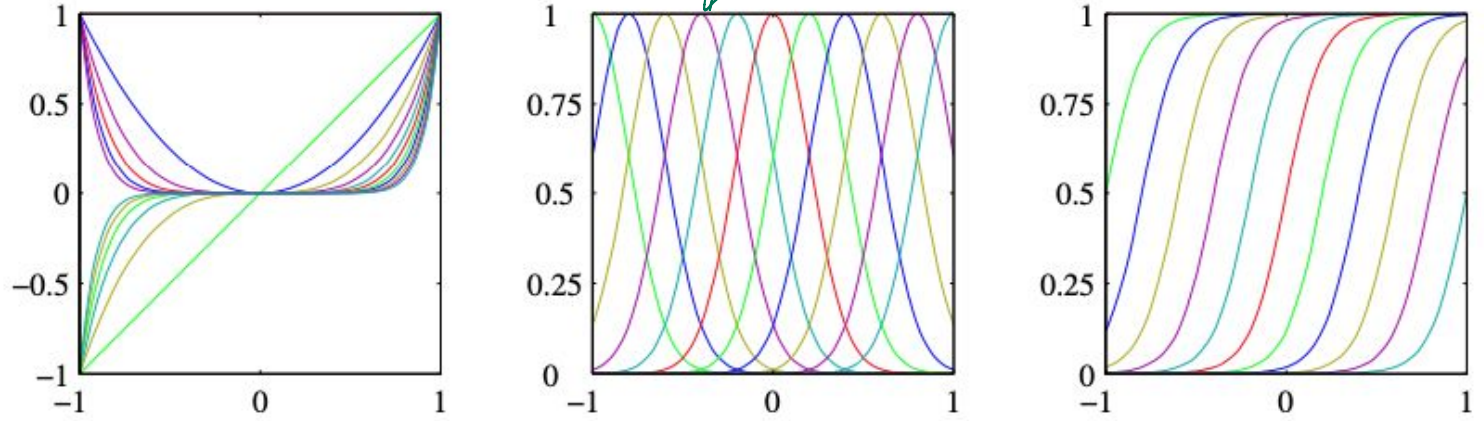


Figure 3.1 Examples of basis functions, showing polynomials on the left, Gaussians of the form (3.4) in the centre, and sigmoidal of the form (3.5) on the right.

So far

Basis functions CAN:

- Represent non-linear decision boundaries in the input space.
- Separate classes linearly (in the feature space) for classes that are not linearly separable in the input space.

Basis functions CAN NOT:

- Remove class overlap that already exist in input space.
- Adapt its own shape to data (but neural nets can, “flexible basis functions”)
- Adapt the number of features to data (this lecture and next lecture …)

Where are we going? Two key ideas

$x \in \mathbb{R}^D$
$w \in \mathbb{R}^M$
$y \sim w^T \phi(x)$ "parametric"

- Instead of "summarising" the training data into a set of weights (with fixed length), why not ask the features to adapt to data?
  - Continuity : Mostly targets don't change abruptly.
  - Similarity : Each training pair (input, target) tells us something about the possible targets in the neighbourhood of the input.

  Kernels formalise these ideas

- Nonparametric methods: do not rely on a fixed number of parameters, but rather usually on storing the entire training set (various loose definitions are used here).

"no parameters"
OR unbounded # of "parameters"

How to get there?

- 0) Histograms
- 1) Simple density estimation kernels (must only be non-negative integrable)
- 2) Implicit feature mapping kernels (must be positive definite)

Warning:

- The term kernel is highly overloaded.
- Even today we consider two different types of kernel:
  1. Smoothing kernel / density estimation kernel / Parzen window estimator kernel / Nadaraya-Watson kernel
  2. Positive semidefinite kernel / reproducing kernel hilbert space kernel / implicit feature map kernel / support vector machine kernel / $\approx$ Gaussian process covariance function or kernel

Density estimation

Observe data points $\{x_n\}_{n=1}^{N}$, draw i.i.d. from p(x)

Goal: estimate p(x) from data

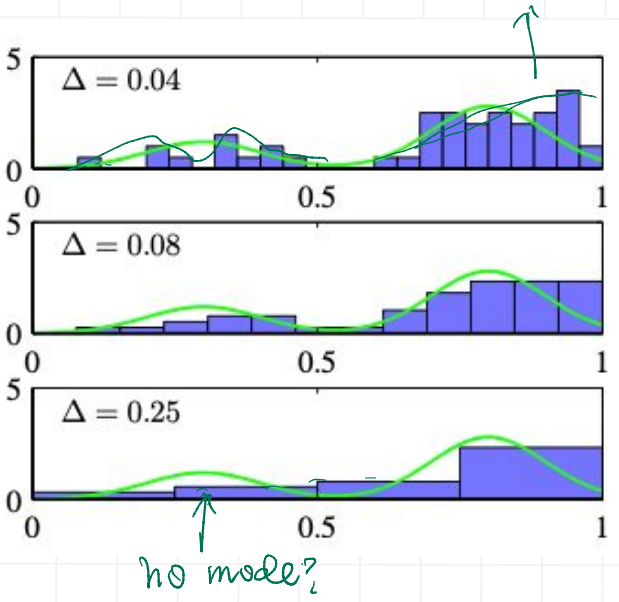


Figure 2.24 An illustration of the histogram approach to density estimation, in which a data set of 50 data points is generated from the distribution shown by the green curve. Histogram density estimates, based on (2.241), with a common bin width Δ are shown for various values of Δ.

Histograms:

- Partition space into bins of width **Δ**
- Count the number of points in each bin
- Normalise

$$p_i = \frac{n_i}{N\Delta_i} \qquad (2.241)$$

Histograms as density estimators

Pros

- Each data point is used once. Applicable to sequentially arriving data.
- Data points can the thrown away once the histogram is computed -- fixed storage cost given bin width.
- Good for quickly visualising densities in 1 or 2 dimensions

what does a 2-dim histogram look like?

Cons

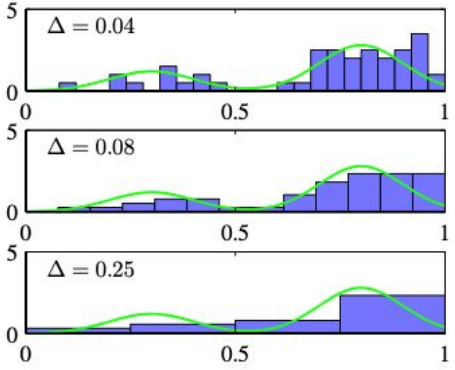
- Depend on ~~bin width Δ~~
- Has discontinuities due to bin edges
- Does not work well in >2 dimensions - curse of dimensionality

Implicit assumption: a distance measure in some local neighbourhood

→

Kernel density estimators
Nearest-neighbour methods

Figure 2.24 An illustration of the histogram approach to density estimation, in which a data set of 50 data points is generated from the distribution shown by the green curve. Histogram density estimates, based on (2.241), with a common bin width Δ are shown for various values of Δ.



Non-parametric density estimation

N data points drawn i.i.d. from unknown distribution p(x) in R^D

Probability mass in a small region $\mathcal{R}$

$$P = \int_{\mathcal{R}} p(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{2.242}$$

Prob that K points fall within $\mathcal{R}$

$$\mathrm{Bin}(K|N,P) = \frac{N!}{K!(N-K)!} P^K (1-P)^{N-K}. \tag{2.243}$$

For large N

$$K \simeq NP. \tag{2.244}$$

$$P \simeq p(\mathbf{x})V \tag{2.245}$$

$$p(\mathbf{x}) = \frac{K}{NV}. \tag{2.246}$$

- Volume V of $\mathcal{R}$ sufficiently small, s.t. p(x) approximately constant in $\mathcal{R}$
- Volume V of $\mathcal{R}$ sufficiently large, s.t. K is sufficiently large and binomial sharply peaked

→ Fix K - K-nearest neighbours
Fix V - kernel density estimation

It can be shown that both the K-nearest-neighbour density estimator and the kernel density estimator converge to the true probability density in the limit N → ∞ provided V shrinks suitably with N , and K grows with N (Duda and Hart, 1973).

Kernel density estimation (KDE), aka Parzen windows

$$p(\mathbf{x}) = \frac{K}{NV}. \tag{2.246}$$

Hypercubes

$$k(\mathbf{u}) = \begin{cases} 1, & |u_i| \leqslant 1/2, \\ 0, & \text{otherwise} \end{cases} \qquad i = 1, \ldots, D, \tag{2.247}$$

$$K = \sum_{n=1}^{N} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right). \tag{2.248}$$

how many points are $\pm \frac{1}{2}h$ within x_n

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right) \tag{2.249}$$

N · volume · k

! We haven't improved from histograms (yet)

Kernel density estimation with Gaussians

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{1/2}}\exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_n\|^2}{2h^2}\right\} \qquad (2.250)$$

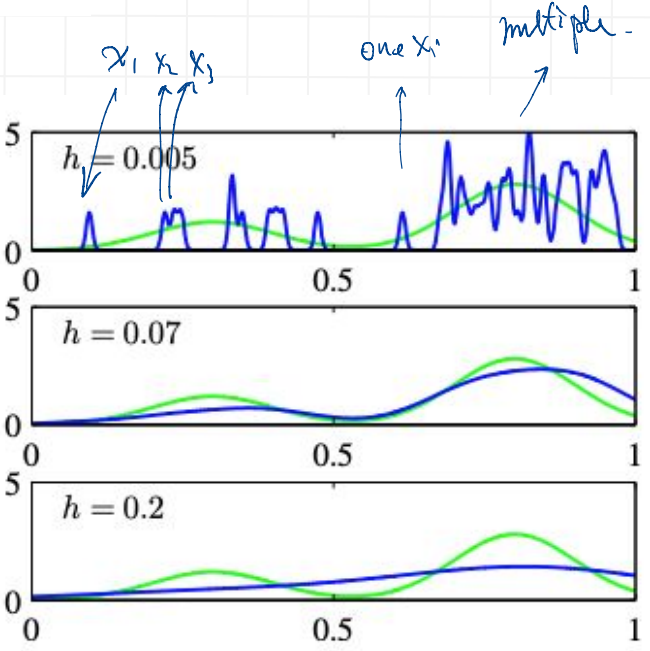


Figure 2.25 Illustration of the kernel density model (2.250) applied to the same data set used to demonstrate the histogram approach in Figure 2.24. We see that h acts as a smoothing parameter and that if it is set too small (top panel), the result is a very noisy density model, whereas if it is set too large (bottom panel), then the bimodal nature of the underlying distribution from which the data is generated (shown by the green curve) is washed out. The best density model is obtained for some intermediate value of h (middle panel).

Kernel density estimation in general

Choose any k(**u**) s.t.

$$k(\mathbf{u}) \geqslant 0, \tag{2.251}$$

$$\int k(\mathbf{u})\,\mathrm{d}\mathbf{u} = 1 \tag{2.252}$$

Pro: training data simply stored, no computation needed for “training”

Con: training data simply stored, expensive to evaluate the density

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right) \tag{2.249}$$

Nearest neighbour methods

$$p(\mathbf{x}) = \frac{K}{NV}. \tag{2.246}$$

Drawback of KDE: fixed V (not adapting to data), fixed k(**u**), having to choose h (1-d kernel width parameter)

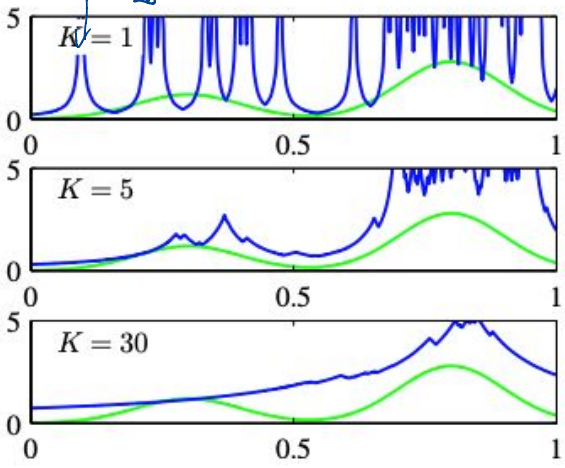
Alternative: fix K, use data to find V

Consider a small sphere around x and then allow the radius to increase until it contains exactly *K* data points. Set V to the volume of the resulting sphere.

! Do not produce true density estimates.

$V \to 0$ when $X \to X_n$, $n = 1, \cdots, N$ $p(x) \to \infty$

Figure 2.26 Illustration of K-nearest-neighbour density estimation using the same data set as in Figures 2.25 and 2.24. We see that the parameter K governs the degree of smoothing, so that a small value of K leads to a very noisy density model (top panel), whereas a large value (bottom panel) smoothes out the bimodal nature of the true distribution (shown by the green curve) from which the data set was generated.



Nearest neighbour for classification

$$p(\mathbf{x}) = \frac{K}{NV}. \tag{2.246}$$

draw a sphere centred on x containing precisely K points irrespective of their class.

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V}. \tag{2.253}$$

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}. \tag{2.256}$$

K = 1 : the *nearest-neighbour* rule

in the limit N → ∞, the error rate of 1-NN is never more than twice the minimum achievable error rate of an optimal classifier, i.e., one that uses the true class distributions (Cover and Hart, 1967) .

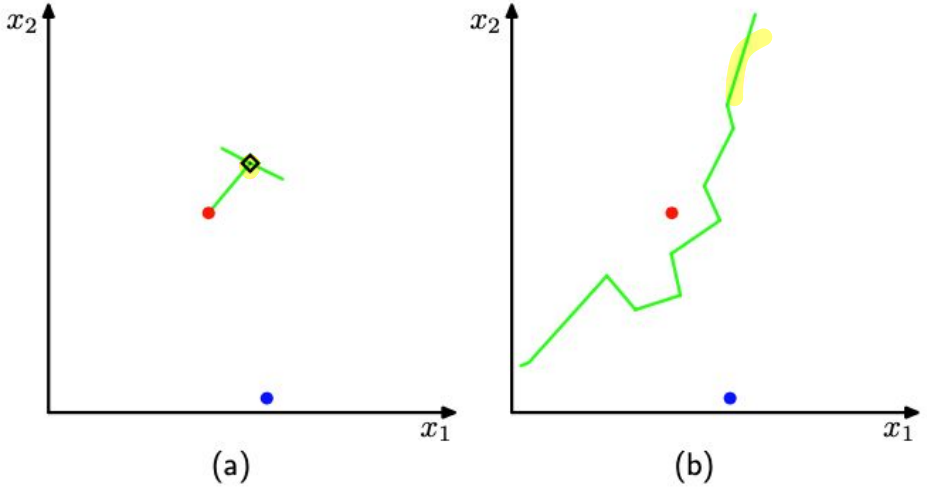


Figure 2.27 (a) In the K-nearest-neighbour classifier, a new point, shown by the black diamond, is classified according to the majority class membership of the K closest training data points, in this case $K = 3$. (b) In the nearest-neighbour ($K = 1$) approach to classification, the resulting decision boundary is composed of hyperplanes that form perpendicular bisectors of pairs of points from different classes.

larger K ≠ better classification

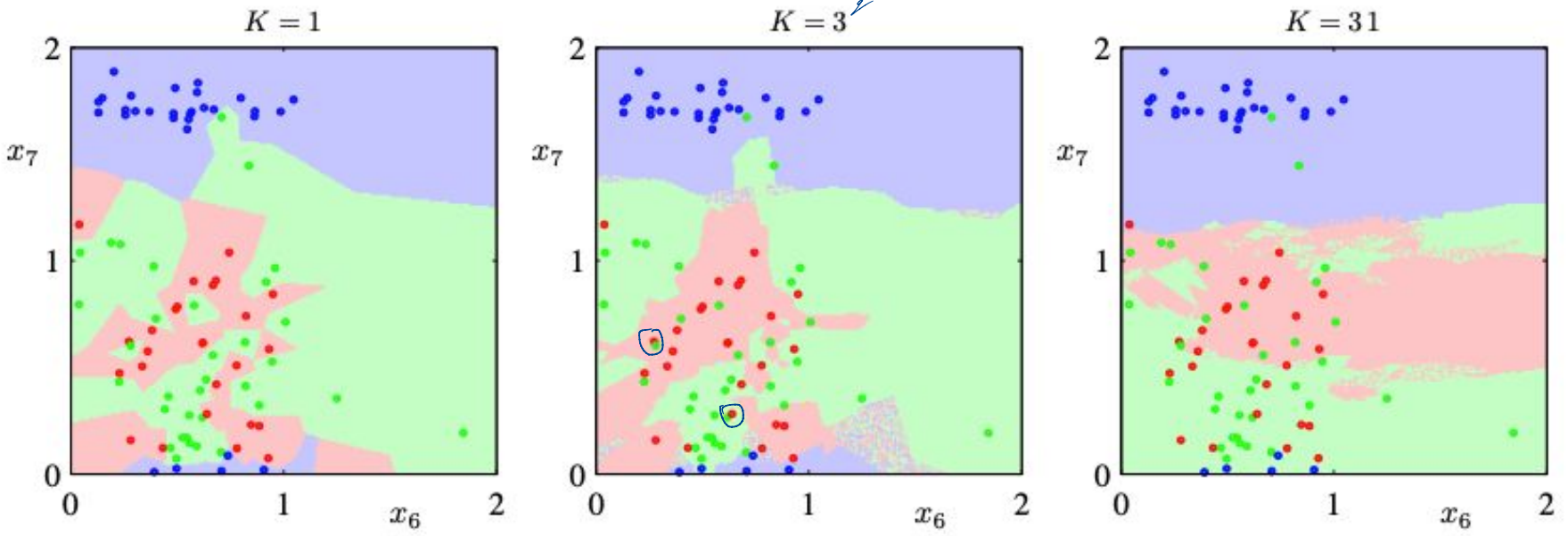


Figure 2.28 Plot of 200 data points from the oil data set showing values of x_6 plotted against x_7, where the red, green, and blue points correspond to the 'laminar', 'annular', and 'homogeneous' classes, respectively. Also shown are the classifications of the input space given by the K-nearest-neighbour algorithm for various values of K.

Kernels

Basis functions recap

Non-parametric methods (Chap 2.5)

Dual representation (of linear models, Chap 6.1)

Constructing kernels (Chap 6.2)

The role of training data

- Parametric methods
  - Learn the model parameter $\mathbf{w}$ from the training data $\mathbf{t}$.
  - Discard the training data $\mathbf{t}$.
- Nonparametric methods
  - Use training data directly for prediction
  - k-nearest neighbours : use k-closest data from the 'training' set for classification
- Kernel methods
  - Base prediction on linear combination of kernel functions evaluated at the training data.

$w^T x$, $w^T \phi(x)$

circa 2005 ~ 2015
"non-parametric Bayesian"
param → ∞,

Input vs Features

- A feature is a measurable property of a phenomenon being observed or any derived property thereof
  - raw features: the original data → input
  - derived features: mappings of the original features to some other space (possibly high- or infinite dimensional, e.g., basis functions)
- Feature selection: which features matter for the problem at hand?
  - redundant features → w.r.t each other
  - problem dependent → are they informative of the target? $I(X_i; t) = 0$.
- Feature extraction: can we combine the important features to a smaller set of new features?
  - compact representation versus ability to explain to a human

Kernel methods in one slide

- Consider a labelled training set $\{\mathbf{x}_i, t_i\}_{i=1}^N$
- On a new point $\mathbf{x}$, we will predict

$$y(\mathbf{x}) = \sum_{i=1}^{N} a_i \cdot K(\mathbf{x}, \mathbf{x}_i)$$

where $\{a_i\}_{i=1}^N$ are weights to be determined based on our training set, and $K(\cdot, \cdot)$ is a kernel function

- This is a major departure from the linear models considered previously!
- The kernel function measures the similarity between any two examples
  - Prediction is a weighted average of the training targets
  - Weights depend on the similarity of $\mathbf{x}$ to each training example

From feature function to kernels

goal → $y(\mathbf{x}) = \sum_{i=1}^{N} a_i \cdot K(\mathbf{x}, \mathbf{x}_i)$

$N \times N$ Gram matrix $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$

$k \sim \text{sim}(x_i, x_j)$

MSE + L2 regularisation -

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} \qquad (6.2)$$

also (3.27)

$$\mathbf{\Phi} = \underset{N\times M}{\begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix}} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^\top \\ \boldsymbol{\phi}(\mathbf{x}_2)^\top \\ \cdots \\ \boldsymbol{\phi}(\mathbf{x}_N)^\top \end{bmatrix}$$

$\mathbf{\Phi}\mathbf{\Phi}^\top \in R^{N\times N}$

$\mathbf{\Phi}^\top\mathbf{\Phi} \in R^{M\times M}$

Optimal regularised w $\quad \mathbf{w}^\star = (\lambda\mathbf{I} + \mathbf{\Phi}^\top\mathbf{\Phi})^{-1}\mathbf{\Phi}^\top\mathbf{t}$

← not what we need

← from chap 3

Prediction function for new **x** $\quad y(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top\mathbf{w}^\star = \boldsymbol{\phi}(\mathbf{x})^\top(\lambda\mathbf{I} + \mathbf{\Phi}^\top\mathbf{\Phi})^{-1}\mathbf{\Phi}^\top\mathbf{t}$

Dual representation

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^\top \\ \boldsymbol{\phi}(\mathbf{x}_2)^\top \\ \cdots \\ \boldsymbol{\phi}(\mathbf{x}_N)^\top \end{bmatrix}$$

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} \qquad (3.27)\ (6.2)$$

$N \times N$ Gram matrix $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$

not $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{t} - \boldsymbol{\Phi}\mathbf{w})^\top(\mathbf{t} - \boldsymbol{\Phi}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}$$

$$\mathbf{w}^\star = (\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{t} \qquad K^\top = K$$

$\frac{\partial J}{\partial w} = 0$

$$(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I})\mathbf{w} = \boldsymbol{\Phi}^\top\mathbf{t}$$

$$\lambda\mathbf{w} = \boldsymbol{\Phi}^\top(\mathbf{t} - \boldsymbol{\Phi}\mathbf{w})$$

$$\mathbf{w} = \boldsymbol{\Phi}^\top\mathbf{a} = \sum_{n=1}^{N}\boldsymbol{\phi}(\mathbf{x}_n)a_n$$

$\vec{a} \in \mathbb{R}^N \qquad \vec{w} \in \mathbb{R}^m$

$$a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}$$

pred. error on x_n

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\mathbf{a} - \mathbf{a}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\mathbf{t} + \frac{1}{2}\mathbf{t}^\top\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\mathbf{a}$$

Dual representation (contd)

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} \tag{6.5}$$

Define the $N \times N$ Gram matrix $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$ with elements

$$K_{nm} = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) \tag{6.6}$$

for x not in $\{x_i\}_{i=1}^{N}$

$$k(x, x') = \phi(x)^{\mathrm{T}}\phi(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x') \tag{6.10}$$

feat. of training point

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}. \tag{6.7}$$

const

linear in α

linear in k

quadratic in α

Solving for a, prediction function

So what is $K(x, x_i)$
① explicitly work it out using $\phi(x)$
② $\phi(x)$ is implicit $K(x_i, x_j)$ implies $\phi(x)$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}. \tag{6.7}$$

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}. \tag{6.8}$$

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

GOAL

$$y(\mathbf{x}) = \sum_{i=1}^{N} a_i \cdot K(\mathbf{x}, \mathbf{x}_i)$$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t} \tag{6.9}$$

$$k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x}).$$

sim. of x w.r.t. $\{x_n\}_{n=1}^N$

pair-wise sim of training $\{x_n\}_{n=1}^N$

inverse of $R^{N\times N}$

pre-compute during training R^N

Prediction function for new **x**

$$y(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^{\top}\mathbf{w}^{\star} = \boldsymbol{\phi}(\mathbf{x})^{\top}(\lambda\mathbf{I} + \mathbf{\Phi}^{\top}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\top}\mathbf{t}$$

training points summarized in $(\lambda I + \Phi^T\Phi)^{-1}\Phi^T t$

Kernels

Basis functions recap

Non-parametric methods (Chap 2.5)

Dual representation (of linear models, Chap 6.1)

Constructing kernels (Chap 6.2)

The kernel function

- The kernel function is defined over two points, $\mathbf{x}$ and $\mathbf{x}'$, of the input space

$$k(x, x') = \phi(x)^{\mathrm{T}} \phi(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x') \tag{6.10}$$

- $k(\mathbf{x}, \mathbf{x}')$ is symmetric.
- It is an inner product of two vectors of basis functions

$$k(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle.$$

- For prediction, the kernel function will be evaluated at the training data points. (See next slides.)

An example

$$k(x, x') = \phi(x)^{\mathrm{T}}\phi(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x') \tag{6.10}$$

inner prod.

$$k(\mathbf{x}, \mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2. \tag{6.11}$$

(x_1, x_2) (z_1, z_2)

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^{\mathrm{T}} \\ &= \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{z}). \end{aligned} \tag{6.12}$$

only involve (x_1, x_2)

only involve (z_1, z_2)

$\phi''(x) = (x_1 \; x_2 \; x_1 x_2 \; x_1^2 \; x_2^2 \; 1)$

$\phi'(x) = (x_1^2 \; x_1 x_2 \; x_2^2)^T$

$$\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^{\mathrm{T}}$$

features of polynomial order 2

More generally, however, we need a simple way to test whether a function constitutes a valid kernel without having to construct the function $\phi(\mathbf{x})$ explicitly. A necessary and sufficient condition for a function $k(\mathbf{x}, \mathbf{x}')$ to be a valid kernel (Shawe-Taylor and Cristianini, 2004) is that the Gram matrix $\mathbf{K}$, whose elements are given by $k(\mathbf{x}_n, \mathbf{x}_m)$, should be positive semidefinite for all possible choices of the set $\{\mathbf{x}_n\}$. Note that a positive semidefinite matrix is not the same thing as a matrix whose elements are nonnegative.

↘ all eig values ≥ 0

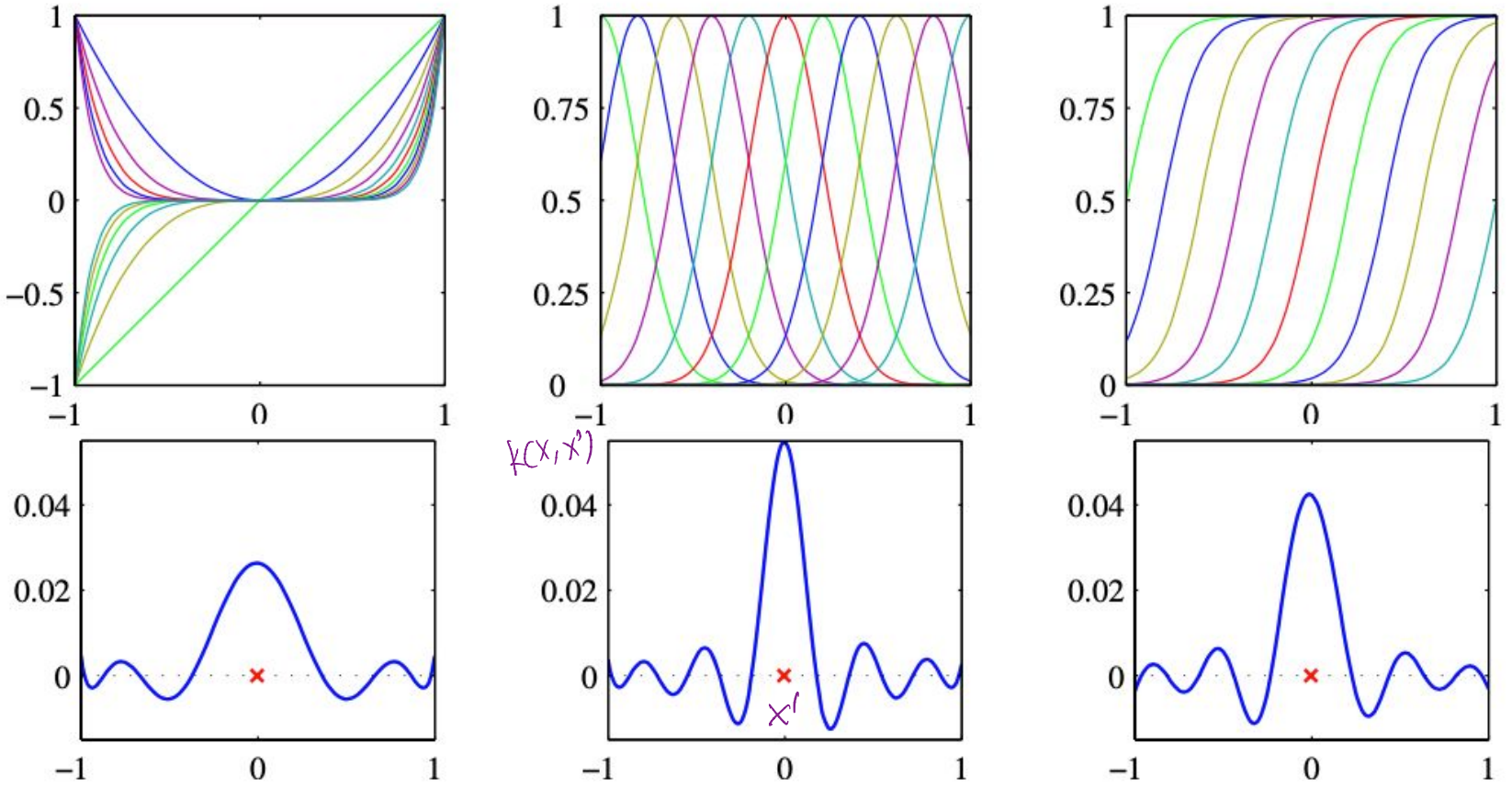


Figure 6.1 Illustration of the construction of kernel functions starting from a corresponding set of basis functions. In each column the lower plot shows the kernel function $k(x, x')$ defined by (6.10) plotted as a function of x for $x' = 0$, while the upper plot shows the corresponding basis functions given by polynomials (left column), 'Gaussians' (centre column), and logistic sigmoids (right column).

Techniques for Constructing New Kernels.

$x^T x'$

$(x^T x')^2$

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = c k_1(\mathbf{x}, \mathbf{x}') \tag{6.13}$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') \tag{6.14}$$

$$k(\mathbf{x}, \mathbf{x}') = q\left(k_1(\mathbf{x}, \mathbf{x}')\right) \tag{6.15}$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \tag{6.16}$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \tag{6.17}$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') \tag{6.18}$$

$$k(\mathbf{x}, \mathbf{x}') = k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) \tag{6.19}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' \tag{6.20}$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \tag{6.21}$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) k_b(\mathbf{x}_b, \mathbf{x}'_b) \tag{6.22}$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Gaussian / Radial Basis Function (RBF) kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right) \tag{6.23}$$

→ what's $\phi(x)$?

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}' \tag{6.24}$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\mathbf{x}^{\mathrm{T}}\mathbf{x}/2\sigma^2\right) \exp\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/\sigma^2\right) \exp\left(-(\mathbf{x}')^{\mathrm{T}}\mathbf{x}'/2\sigma^2\right) \tag{6.25}$$

$f(x)$ $\exp\left(\frac{1}{\sigma^2} k(x, x')\right)$ $f(x')$

$k(x, x')$

$\downarrow$

$x^T x'$

$\phi(x) = x$

Further examples of kernels

$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}')^M$ — only terms of degree M

$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^M$ — all terms up to degree M

easier to compute than $\phi(x)^\top \phi(x')$

$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right)$ — Gaussian kernel

$k(\mathbf{x}, \mathbf{x}') = \tanh\left(a\,\mathbf{x}^\top \mathbf{x}' + b\right)$ — Sigmoidal kernel (invalid)

Generally, we call

$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ — linear kernel

$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$ — stationary kernel (only depend on diff)

$k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$ — homogeneous kernel (only depend on distance)

Kernels over graphs, strings, sets

- We 'only' need an appropriate similarity measure $k(\mathbf{x}, \mathbf{x}')$ which is a kernel.
- Example: Given a set $\mathcal{A}$ and the set of all subsets of $\mathcal{A}$, called the power set $\mathcal{P}(\mathcal{A})$.
- For two subsets $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{P}(\mathcal{A})$, denote the number of elements of the intersection of $\mathcal{A}_1$ and $\mathcal{A}_2$ by $|\mathcal{A}_1 \cap \mathcal{A}_2|$.
- Then it can be shown that

$$k(\mathcal{A}_1, \mathcal{A}_2) = 2^{|\mathcal{A}_1 \cap \mathcal{A}_2|}$$

corresponds to an inner product in a feature space. Therefore, $k(\mathcal{A}_1, \mathcal{A}_2)$ is a valid kernel function.

Kernels from probabilistic generative models

- Given $p(\mathbf{x})$, we can define a kernel

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})\, p(\mathbf{x}'),$$

  which means two inputs $\mathbf{x}$ and $\mathbf{x}'$ are similar if they both have high probabilities.

- Include a weighting function $p(i)$ and extend the kernel to

$$k(\mathbf{x}, \mathbf{x}') = \sum_i p(\mathbf{x} \mid i)\, p(\mathbf{x}' \mid i) p(i).$$

- For a continous variable $\mathbf{z}$

$$k(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{x}' \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}.$$

- Hidden Markov Model with sequences of length L.

Kernels for regression and classification: summary

- Pick a suitable kernel function $k(\mathbf{x}, \mathbf{x}')$
  - e.g. by computing inner product of some basis functions
- Make predictions by suitably combining $k(\mathbf{x}, \mathbf{x}_n)$ for each training example $\mathbf{x}_n$
  - implicitly, a linear model in some high-dimensional space
- For linear regression, we go from

$$y(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top (\lambda \mathbf{I} + \boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{t}$$

$\rightarrow w^*$

to

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

$\rightarrow \vec{a}$

  - can plug in suitable kernel function to implicitly perform nonlinear transformation

no need to write out $\phi(x)$

Cons of $k(x, x')$

$K^{N \times N}$ $O(N^2)$

$(K + \lambda I_N)^{-1}$ $O(N^3)$

benefits of using $k(x, x')$

- $\phi(x)$ can be implicitly "infinite" dimensions
- compose new k from existing k
- "sparse" $\vec{a}$
- RBF SVMs → "sparse" weighted nearest neighbour
- New toolset

Kernels

Basis functions recap

Non-parametric methods (Chap 2.5)

Dual representation (of linear models, Chap 6.1)

Constructing kernels (Chap 6.2)