# Overview for Week 4

## Linear and Nonlinear Functions

Function Classes, Feature Extraction, Dot Products

## Polynomial Kernels

Features, Explicit Representation

## Radial Basis Function Kernels

Gaussian, Laplacian, admissibility criteria

## Kernel Regression

Equations, Algorithm, Regularization

## Regularization and Operators

Smoothness Criteria, Connection to Kernels, Fourier Representation

## Linear Functions

THE AUSTRALIAN NATIONAL UNIVERSITY

### Linear Function

Denote by $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Then a linear function is given by

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

### Generalized Linear Function

Denote by $f_1, \ldots, f_n$ $n$ arbitrary functions $\mathcal{X} \longrightarrow \mathbb{R}$, then a generalized linear function is given by

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i f_i(\mathbf{x})$$

### Connection

Set $n = d + 1$, $f_i(\mathbf{x}) = x_i$ (coordinate projections), and $f_{d+1} = 1$. Then Linear Functions are a special case of Generalized Linear Functions.

## Things you can do with Linear Functions

THE AUSTRALIAN NATIONAL UNIVERSITY

### Linear Estimators

We find a linear mapping from $\mathcal{X}$ into $\mathbb{R}$ (of course, that's the most boring applications).

### Hyperplanes

A linear function defines a hyperplane $H$ by

$$H := \{\mathbf{x} | \langle \mathbf{w}, \mathbf{x} \rangle = c\}$$

This is the set of all points with distance $\frac{c}{\|\mathbf{w}\|}$ from the origin along the vector $\mathbf{w}$.

Proof: for point along the line $\alpha\mathbf{w}$ with $\langle \alpha\mathbf{w}, \mathbf{w} \rangle = c$ we have $\alpha = \frac{c}{\|\mathbf{w}\|^2}$

$$d(0, \alpha\mathbf{w})^2 = \|\alpha\mathbf{w}\|^2 = |\alpha|^2 \langle \mathbf{w}, \mathbf{w} \rangle = \frac{c^2}{\|\mathbf{w}\|^2}$$

### Half-Spaces

Likewise we can define spaces by

$$H := \{\mathbf{x} | \langle \mathbf{w}, \mathbf{x} \rangle \leq c\} \text{ or } H := \{\mathbf{x} | \langle \mathbf{w}, \mathbf{x} \rangle \leq c\}$$

## Feature Extraction

THE AUSTRALIAN NATIONAL UNIVERSITY

### Idea

Sometimes problems may not be linearly separable, yet a simple solution may exist in terms of some specific features (a possible way of inserting prior knowledge).

### Example

The number of loops in a character can be used to distinguish between $0, 1, 8$, the number of intersections will help us to deal with 4 and 6, too, $\ldots$

### Nonlinearization

We use a feature extractor $\Phi : \mathfrak{X} \to \mathfrak{F}$ with $\mathbf{x} \to \Phi(\mathbf{x})$. So all we have to do is **replace every occurence of x by** $\Phi(\mathbf{x})$ to obtain a nonlinear estimator. We get $f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b$ and for $\mathbf{w} = \sum_{i=1}^{m} \alpha_i \Phi(\mathbf{x}_i)$ this leads to

$$f(\mathbf{x}) = \left\langle \sum_{i=1}^{m} \alpha_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \right\rangle + b = \sum_{i=1}^{m} \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + b$$

## Examples

THE AUSTRALIAN NATIONAL UNIVERSITY

### Two Interlocking Spirals

If we transform the data $(x_1, x_2)$ into a radial part ($r = \sqrt{x_1^2 + x_2^2}$) and an angular part ($x_1 = r \cos \phi$, $x_1 = r \sin \phi$), the problem becomes much easier to solve (we only have to distinguish different stripes).

### Japanese Character Recognition

Break down the images into strokes and recognize it from the latter (there's a pre-defined order which can be useful).

### Medical Diagnosis

Include physician's comments, knowledge about unhealthy combinations, detect the presence of certain features (EEG), . . .

### Suitable Rescaling

If we observe, say the weight and the height of a person (e.g. measured in centimeters or in meters), typically rescaling with zero mean and unit variance is a good idea.

# The Polynomial Features ride again

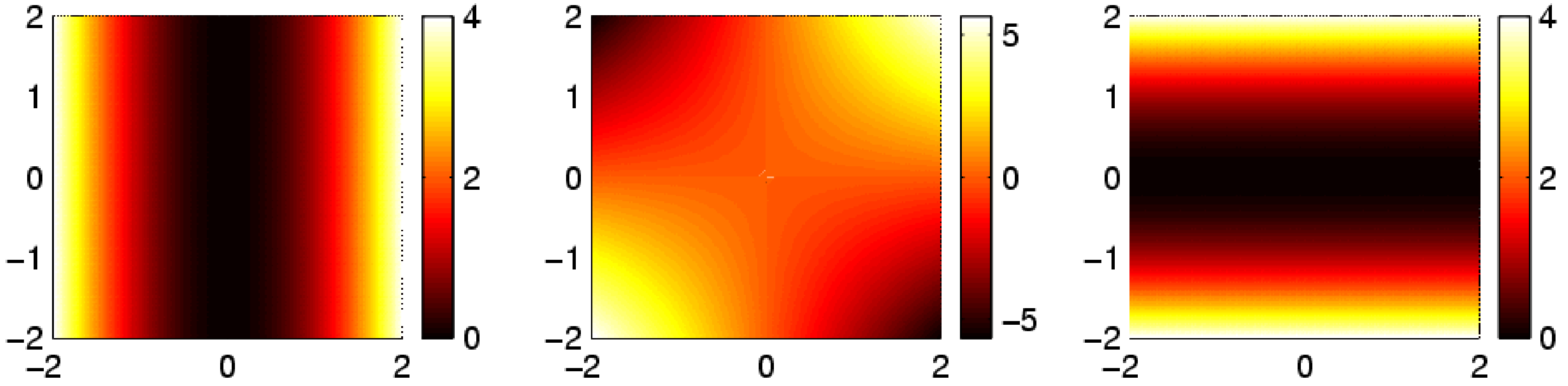
## Quadratic Features in $\mathbb{R}^2$

$$\Phi(x) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

## Dot Product

$$\langle \Phi(x), \Phi(x') \rangle = \left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left({x'_1}^2, \sqrt{2}x'_1x'_2, {x'_2}^2\right) \right\rangle = \langle x, x' \rangle^2.$$

This trick does not only work for 2nd order polynomials but for any $\langle x, x' \rangle^d$.



## Kernels

### Problem

Extracting features can sometimes be very costly. Just imagine computing second order features in 1000 dimensions. This leads to 5005 numbers, for higher order polynomial features this is even way worse.

### Solution

Don't compute the features, try to compute dot products implicitly. For some features this works ... and those are the ones we use.

### Definition

A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric function in its arguments for which the following property holds

$$k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \text{ for some feature map } \Phi.$$

The hope is that $k(\mathbf{x}, \mathbf{x}')$ will be much cheaper to compute than $\Phi(\mathbf{x})$.

# Kernel Matrix

## Basic Idea

To compare observations we compute dot products (recall the perceptron algorithms). There the dot product matrix $K$ given by $K_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. This is transformed into

$$K_{ij} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle = k(\mathbf{x}_i, \mathbf{x}_j)$$

where $\mathbf{x}_i$ are the training patterns $\mathbf{x}_i$.

## Similarity Measure

The entries $K_{ij}$ tell us the amount of overlap between $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$, so, in a sense, $k(\mathbf{x}_i, \mathbf{x}_j)$ is a similarity measure.

## Distance in Feature Space

We get a distance measure between points in feture space $\Phi(\mathbf{x})$ and $\Phi(\mathbf{x}')$ via

$$\begin{aligned} d(\mathbf{x}, \mathbf{x}')^2 &:= \|\Phi(\mathbf{x}) - \Phi(\mathbf{x}')\|^2 = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}) \rangle - 2\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle + \langle \Phi(\mathbf{x}'), \Phi(\mathbf{x}') \rangle \\ &= k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}, \mathbf{x}) \end{aligned}$$

# Kernel Matrix, Part II

## The Kernel Matrix is Positive Semidefinite

We have $\alpha^\top K\alpha \geq 0$ for all $\alpha \in \mathbb{R}^m$ and all kernel matrices $K \in \mathbb{R}^{m\times m}$. This is so since

$$\begin{aligned}\sum_{i,j}^m \alpha_i\alpha_j K_{ij} &= \sum_{i,j}^m \alpha_i\alpha_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle \\ &= \left\langle \sum_i^m \alpha_i\Phi(\mathbf{x}_i), \sum_j^m \alpha_j\Phi(\mathbf{x}_j)\right\rangle = \left\| \sum_{i=1}^m \alpha_i\Phi(\mathbf{x}_i)\right\|^2 \geq 0\end{aligned}$$

## Kernel Expansion

For a function given by $f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x})\rangle + b$ with $\mathbf{w} = \sum_{i=1}^m \alpha_i\Phi(\mathbf{x}_i)$ we obtain

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x})\rangle + b = \left\langle \sum_{i=1}^m \alpha_i\Phi(\mathbf{x}_i), \Phi(\mathbf{x})\right\rangle + b = \sum_{i=1}^m \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b.$$

In particular for the function values $\mathbf{f} := (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m))$ we have $\mathbf{f} = K\alpha$.

# Polynomial Kernels in $\mathbb{R}^n$

THE AUSTRALIAN NATIONAL UNIVERSITY

## Idea

We want to extend $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^2$ to

$$k(\mathbf{x}, \mathbf{x}') = \left(\langle \mathbf{x}, \mathbf{x}' \rangle + c\right)^d \text{ where } c > 0 \text{ and } d \in \mathbb{N}.$$

For this purpose we have to prove that such a kernel corresponds to a dot product.

## Proof Strategy

Stupid and straightforward: we simply compute the explicit sum given by the kernel, i.e.

$$k(\mathbf{x}, \mathbf{x}') = \left(\langle \mathbf{x}, \mathbf{x}' \rangle + c\right)^d = \sum_{i=0}^{m} \binom{d}{i} \left(\langle \mathbf{x}, \mathbf{x}' \rangle\right)^i c^{d-i}$$

The next step is to show that the individual terms $\left(\langle \mathbf{x}, \mathbf{x}' \rangle\right)^i$ are dot products for some $\Phi_i(\mathbf{x})$.

# Polynomial Kernels in $\mathbb{R}^n$, Part II

THE AUSTRALIAN NATIONAL UNIVERSITY

## Notation

To simplify notation we use the following abbreviations concerning $\mathbf{i} \in \mathbb{N}^n$ and $\mathbf{x} \in \mathbb{R}^n$.

$$\begin{aligned}\mathbf{i}! &:= i_1! \cdot i_2! \cdot \ldots \cdot i_n! \\ \mathbf{x}^{\mathbf{i}} &:= x_1^{i_1} \cdot x_2^{i_2} \cdot \ldots \cdot x_n^{i_n} \\ |\mathbf{i}| &:= i_1 + i_2 + \ldots + i_n\end{aligned}$$

Quite often, $\mathbf{i}$ is referred to as a **multi index**.

## Brute Force Expansion

$$\left(\langle \mathbf{x}, \mathbf{x}' \rangle\right)^d = \sum_{|\mathbf{i}|=1} \frac{d!}{\mathbf{i}!} \mathbf{x}^{\mathbf{i}} (\mathbf{x}')^{\mathbf{i}} = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$$

where $\Phi(\mathbf{x}) = \left(\ldots \sqrt{\frac{d!}{\mathbf{i}!}} \mathbf{x}^{\mathbf{i}} \ldots\right)$. This means that we compute dot products cheaply in the space spanned by all possible polynomial features.

## Are all $k(\mathbf{x}, \mathbf{x}')$ good Kernels?

**Computability**

We have to be able to compute $k(\mathbf{x}, \mathbf{x}')$ efficiently (much cheaper than dot products themselves).

**"Nice and Useful" Functions**

The features themselves have to be useful for the learning problem at hand. Quite often this means smooth functions.

**Symmetry**

Obviously $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ due to the symmetry of the dot product $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle = \langle \Phi(\mathbf{x}'), \Phi(\mathbf{x}) \rangle$.

**Dot Product in Feature Space**

There has to exist a $\Phi$ such that $k$ really is a dot product.