Machine Learning

Kernel Methods

Dariush Hosseini

dariush.hosseini@ucl.ac.uk
Department of Computer Science
University College London

Lecture Overview

Lecture Overview

By the end of this lecture you should:

1. Know that **Kernel Methods** provide a mechanism for tractably accessing non-linear features for use with linear machine learning methods
2. Understand the relationship between **feature maps** and **kernels** via **Mercer's Theorem**
3. Understand when it is sensible to apply kernel methods via a consideration of the **Representer Theorem**

Lecture Overview

Overview

- Kernel methods seek to **implicitly** map our data into a **higher-dimensional feature space** so that **linear** machine learning algorithms can be used in this feature space
- Why?
  - Linear methods are well understood and often efficient
  - An implicit mapping allows us to access high, even infinite, dimensional feature spaces efficiently

Overview

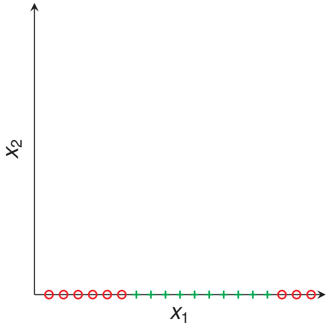
- Usually kernel based methods will follow two steps:

  1. Map the data from the input space, $\mathbb{X}$, to a higher-dimensional **feature space**, $\mathbb{V}$, using some non-linear mapping, $\phi : \mathbb{X} \rightarrow \mathbb{V}$
  2. Run a linear machine learning algorithm in this new space

- This approach has wide applicability - many learning algorithms can be transformed into non-linear high dimensional methods, for example:
  - Ridge Regression
  - Support Vector Machine
  - PCA

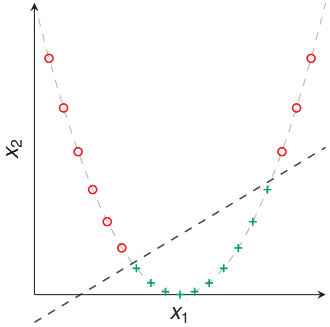
Example

- **Input Data:**
  - $\mathbf{x} = x_1 \in \mathbb{X}$
  - $\mathbb{X} \subseteq \mathbb{R}^1$



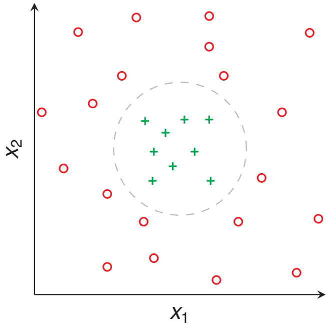
Example

- **Feature Mapped Data:**
  - $\phi(\mathbf{x}) = [x_1, x_1^2]^T \in \mathbb{V}$
  - $\mathbb{V} \subseteq \mathbb{R}^2$



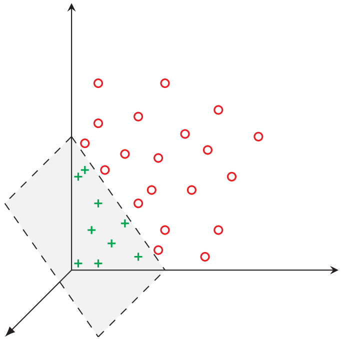
Example

- **Input Data:**
  - $\mathbf{x} = [x_1, x_2]^T \in \mathbb{X}$
  - $\mathbb{X} \subseteq \mathbb{R}^2$



Example

- **Feature Mapped Data:**
  - $\phi(\mathbf{x}) = [x_1^2, \sqrt{2}x_1 x_2, x_2^2]^T \in \mathbb{V}$
  - $\mathbb{V} \subseteq \mathbb{R}^3$



Considerations: Why

- Are we justified in applying this strategy?
- Why should a linear relationship exist in a higher dimensional space?
- **Cover's Theorem** gives some intuition

Cover's Theorem

- Assume that our problem is that of binary classification
- The theorem states that the probability that n data points can be linearly separated in m dimensions is given by:

$$\begin{cases} 1, & n \leqslant m+1 \\ \frac{1}{2^{n-1}} \sum_{i=0}^{m} \begin{pmatrix} n-1 \\ i \end{pmatrix}, & n \geqslant m+1 \end{cases}$$

- Thus the probability of n instances being linearly separable increases with growing dimensionality m

Considerations: How

- How can we operate efficiently in a high dimensional space?
  - For example, consider **matrix inversion** in ridge regression for an infinite dimensional feature space
- We would like to be able to enjoy the benefits of feature mapping without having to perform that mapping or to perform manipulations in elevated dimensions **explicitly**
- The **Kernel Trick** let's us do exactly this

Kernel Trick

- Consider two input points, $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{X}$
- Consider a feature mapping, $\phi : \mathbb{X} \to \mathbb{V}$
- We can define an **inner product** on $\mathbb{V}$, via the **kernel function**, κ, such that:

$$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)}) \cdot \phi(\mathbf{x}^{(j)})$$

Kernel Trick

- Let us consider the description of our learning algorithm for a moment:
- *'If an algorithm is described solely in terms of inner products, $\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$, in input space, then it can be lifted into feature space by replacing the occurrence of those inner products by $\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$'* [Rasmussen & Williams (2006)]
- But what kind of algorithm looks like that?

Aside: Ridge Regression

- Recall the optimisation problem associated with Ridge Regression:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} \right)^2 + \lambda \|\mathbf{w}\|_2^2$$

- This is equivalent to:

$$\begin{aligned} &\min_{\mathbf{w}} && \sum_{i=1}^{n} \xi^{(i)2} + \lambda \|\mathbf{w}\|_2^2 \\ &\text{subject to:} && \xi^{(i)} = y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} \end{aligned} \tag{1}$$

Aside: Ridge Regression & Lagrange Duality

- From the Appendix we see this problem can be solved by seeking the solution to its dual optimisation problem

- First we write the Lagrangian for problem (1):

$$\mathcal{L}(\boldsymbol{\xi}, \mathbf{w}, \widetilde{\boldsymbol{\alpha}}) = \sum_{i=1}^{n} \xi^{(i)2} + \sum_{i=1}^{n} \widetilde{\alpha}^{(i)} \left(y^{(i)} - \xi^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} \right) + \lambda \|\mathbf{w}\|_2^2$$

  Here $\widetilde{\boldsymbol{\alpha}} = [\widetilde{\alpha}^{(1)}, ..., \widetilde{\alpha}^{(n)}]$ are Lagrange Multipliers

- The dual objective can be written:

$$\mathcal{D}(\widetilde{\boldsymbol{\alpha}}) = \min_{\mathbf{w}, \boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi}, \mathbf{w}, \widetilde{\boldsymbol{\alpha}})$$

Aside: Ridge Regression & Lagrange Duality

- This is an unconstrained optimisation which we can solve by seeking stationary points:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = -\sum_{i=1}^{n} \widetilde{\alpha}^{(i)} \mathbf{x}^{(i)} + 2\lambda \mathbf{w}^* = 0 \quad \Longrightarrow \quad \mathbf{w}^* = \frac{1}{2\lambda} \sum_{i=1}^{n} \widetilde{\alpha}^{(i)} \mathbf{x}^{(i)}$$

$$\frac{\partial \mathcal{L}}{\partial \xi^{(i)}} = 2\xi^{(i)*} - \widetilde{\alpha}^{(i)} = 0 \quad \Longrightarrow \quad \xi^{(i)*} = \frac{\widetilde{\alpha}^{(i)}}{2}$$

Aside: Ridge Regression & Lagrange Duality

- Substituting these expressions back into $\mathcal{D}(\widetilde{\boldsymbol{\alpha}})$ yields:

$$
\begin{aligned}
\mathcal{D}(\widetilde{\boldsymbol{\alpha}}) &= \frac{1}{4}\sum_{i=1}^{n}\widetilde{\alpha}^{(i)2} + \sum_{i=1}^{n}\widetilde{\alpha}^{(i)}y^{(i)} - \frac{1}{2}\sum_{i=1}^{n}\widetilde{\alpha}^{(i)2} \\
&\qquad - \frac{1}{2\lambda}\sum_{i,j=1}^{n}\widetilde{\alpha}^{(i)}\widetilde{\alpha}^{(j)}\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)} + \lambda\left(\frac{1}{4\lambda^2}\left\|\sum_{i=1}^{n}\widetilde{\alpha}^{(i)}\mathbf{x}^{(i)}\right\|_2^2\right) \\
&= -\frac{1}{4}\sum_{i=1}^{n}\widetilde{\alpha}^{(i)2} + \sum_{i=1}^{n}\widetilde{\alpha}^{(i)}y^{(i)} - \frac{1}{4\lambda}\sum_{i,j=1}^{n}\widetilde{\alpha}^{(i)}\widetilde{\alpha}^{(j)}\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)} \\
&= -\lambda^2\sum_{i=1}^{n}\alpha^{(i)2} + 2\lambda\sum_{i=1}^{n}\alpha^{(i)}y^{(i)} - \lambda\sum_{i,j=1}^{n}\alpha^{(i)}\alpha^{(j)}\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)}
\end{aligned}
$$

where: $\widetilde{\alpha}^{(i)} = 2\lambda\alpha^{(i)}$

Aside: Ridge Regression & Dual Problem

- This leads to the following dual problem:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \quad -\lambda\|\boldsymbol{\alpha}\|^2 + 2\boldsymbol{\alpha}\cdot\mathbf{y} - \boldsymbol{\alpha}^T(\mathbf{X}\mathbf{X}^T)\boldsymbol{\alpha} \\ \text{or:} \quad &\max_{\boldsymbol{\alpha}} \quad -\boldsymbol{\alpha}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{\alpha} + 2\boldsymbol{\alpha}\cdot\mathbf{y} \end{aligned}$$

- The solution to which is:

$$\begin{aligned} \mathbf{w}^* &= \mathbf{X}^T\boldsymbol{\alpha} \\ \boldsymbol{\alpha} &= (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y} \end{aligned}$$

Kernel Trick: Revisited

- Again let us consider the description of our learning algorithm:
- *'If an algorithm is described solely in terms of inner products,* $\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$*, in input space, then it can be lifted into feature space by replacing the occurrence of those inner products by* $\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$*'* [Rasmussen & Williams (2006)]
  - For example, consider the dual problem of Ridge Regression:

$$\max_{\boldsymbol{\alpha}} \quad -\boldsymbol{\alpha}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{\alpha} + 2\boldsymbol{\alpha}\cdot\mathbf{y}$$

$$\implies \max_{\{\alpha^{(i)}\}_{i=1}^n} \quad -\sum_{i,j=1}^{n} \alpha^{(i)}\alpha^{(j)}\mathbf{x}^i \cdot \mathbf{x}^j - \lambda\sum_{i=1}^{n}\alpha^{(i)2} + 2\sum_{i=1}^{n}\alpha^{(i)}y^{(i)}$$

Kernel Trick

- This is the **Kernel Trick**:
  - It means that, for certain problems, we can enjoy the benefits of feature mapping without ever having to calculate $\phi(\mathbf{x})$ explicitly
  - Instead we can calculate the mapping **implicitly** by calculating $\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)}) \cdot \phi(\mathbf{x}^{(j)})$
- In particular, it turns out that for many machine learning algorithms, when their associated optimisation problem is re-cast in its **dual form**, the inner product arises naturally:
  - We shall investigate the characteristics of such problems later on

Lecture Overview

What is a Kernel?

- We now know that we can exploit the kernel trick to replace the inner products in (certain) learning algorithms with a kernel function
- And we would like to be able to work directly with such functions
- But how do we know if a particular function, κ, really has a feature map, ϕ, associated with it such that, $\kappa(\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)}) \cdot \phi(\mathbf{x}^{(j)})$?
- And what characterises a valid function, κ?

Mercer's Theorem

- **Definition**:

  The **Gram Matrix**, $\mathbf{K}$, is an $n \times n$ matrix with elements, $\{\mathbf{K}_{ij} = \kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})\}_{i=1,j=1}^{n,n}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^m$

- **Definition**:

  $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is positive semidefinite (psd) if it is symmetric and if the Gram Matrix, $\mathbf{K}$, is psd for all n and for all $\mathbf{x}^{(i)} \in \mathbb{R}^m$

Mercer's Theorem

- **Mercer's Theorem**:
  A kernel function, κ, is psd if and only if:

  $$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)}) \cdot \phi(\mathbf{x}^{(j)}) \qquad \text{where} \qquad \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^m$$

  For some feature map, $\phi : \mathbb{R}^m \to \mathbb{W}$ (and Hilbert Space $\mathbb{W}$)

- We call a kernel which satisfies this theorem a **Mercer Kernel**

Mercer's Theorem: Aside

- Let us prove the only if:
  - If:
$$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)}) \cdot \phi(\mathbf{x}^{(j)})$$
  - Then:
$$\mathbf{c}^T\mathbf{K}\mathbf{c} = \sum_{i,j=1}^{n} c_i c_j \kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(\sum_{i=1}^{n} c_i \phi(\mathbf{x}^{(i)})\right) \cdot \left(\sum_{j=1}^{n} c_j \phi(\mathbf{x}^{(j)})\right)$$
$$= \|c_i \phi(\mathbf{x}^{(i)}))\|_2^2 \geqslant 0$$

  This holds for all $\mathbf{c} \in \mathbb{R}^n$ where c_i is the i-th element of $\mathbf{c}$

Polynomial Kernel

- The **polynomial kernel** can be used to compute a compact version of the polynomial feature mapping up to degree p:

$$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = (1 + \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)})^p$$

- Let's try expanding for $p = 2$ and $\mathbf{x} \in \mathbb{R}^2$:

$$\begin{aligned}\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) &= (1 + x_1^{(i)} x_1^{(j)} + x_2^{(i)} x_2^{(i)})^2 \\ &= 1 + 2x_1^{(i)} x_1^{(j)} + 2x_2^{(i)} x_2^{(j)} + \left(x_1^{(i)} x_1^{(j)}\right)^2 + \left(x_2^{(i)} x_2^{(j)}\right)^2 + 2x_1^{(i)} x_2^{(i)} x_1^{(j)} x_2^{(j)}\end{aligned}$$

- In other words our use of the kernel is equivalent to a mapping into 6-dimensional feature space:

$$\phi(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2]^T$$

Radial Basis Function Kernel

- The **RBF kernel** is a flexible measure of similarity parameterised by a **bandwidth** hyperparameter, σ^2:

$$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{2\sigma^2}\right)$$

- The RBF kernel actually gives access to an infinite dimensional feature space.

Document Kernel

- Kernels can be constructed for a variety of data types (graphs, categorical data, etc.)
- The **cosine similarity kernel** can be used as a **document kernel** for the **bag-of-words** representation:

$$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}}{\|\mathbf{x}^{(i)}\|_2^2 \|\mathbf{x}^{(j)}\|_2^2}$$

- Here the input attributes are all counts, and so the measure falls in the range $[0, 1]$

Recap

- We now know that:
  - **Mercer’s Theorem** allows us to specify a **feature map** implicitly if we specify a **Mercer kernel**
  - And the **Kernel Trick** makes clear that if our learning problem can be written in terms of dot products of the input vectors then we can move to this kernel-defined feature space by simply replacing the dot products with the corresponding kernel mapped output
  - **...**But how do we know if a particular learning problem can be written in this way**...**
  - ...For this we need to consider the **Representer Theorem**

Lecture Overview

Notation

- **Training Sample**:

$$\{(\mathbf{x}^{(i)}, y^{(i)}) | \mathbf{x}^{(i)} \in \mathbb{X} \subseteq \mathbb{R}^m, y^{(i)} \in \mathbb{R}\}_{i=1}^n$$

- **Feature Map**:

$$\phi : \mathbb{R}^m \to \mathbb{R}^k$$

- **Weight Vector**:

$$\mathbf{w} \in \mathbb{R}^k$$

- **Linear Prediction Function**:

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})$$

- **Loss Function**:

$$\widetilde{L} : (y, f(\mathbf{x})) \mapsto \widetilde{L}(y, f(\mathbf{x})) \in \mathbb{R}$$

- And some function:

$$\Omega : \mathbf{w} \mapsto \Omega(\mathbf{w}) \in \mathbb{R}$$

Representer Theorem: Sketch

- For a regularised loss function, L, defined such that:

$$L(\mathbf{w}) = \sum_{i=1}^{n} \widetilde{L}(y^{(i)}, \mathbf{w} \cdot \phi(\mathbf{x}^{(i)})) + \Omega(\mathbf{w})$$

- If and only if $\Omega(\mathbf{w})$ is a non-decreasing function of $\|\mathbf{w}\|_2^2$ then, if $\mathbf{w}^*$ minimises L, it admits the following representation:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}^{(i)}) \qquad \text{where: } \alpha_i \in \mathbb{R}$$

- And therefore for a novel test point, $\mathbf{z}$:

$$f(\mathbf{z}) = \mathbf{w} \cdot \phi(\mathbf{z}) = \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{x}^{(i)}, \mathbf{z})$$

Representer Theorem

- This is a powerful result
  - For learning algorithms whose optimisation problem satisfies the conditions of the representer theorem...
  - ...We need never directly seek $\mathbf{w}^*$ (whose dimensions are possibly infinite)
  - Instead we need only seek the n parameters which characterise $\boldsymbol{\alpha}$
  - Furthermore, when evaluating a test point we need never explicitly calculate $\phi(\mathbf{z})$
  - Instead we need only sum over a weighted set of n kernel function outputs, evaluated as the contraction of $\mathbf{z}$ and each of the training points $\{\mathbf{x}^{(i)}\}_{i=1}^n$

Representer Theorem: Sparsity

- Note that a **sparsity-inducing** regulariser sich as $\Omega(\mathbf{w}) = \|\mathbf{w}\|_1^1$, as in the **LASSO**, does not satisfy the conditions of the Representer Theorem
- This explains why we cannot use the kernel trick in conjunction with this sort of **feature selection**
- Sparsity comes at a price!

Lecture Overview

Ridge Regression: Primal Form

- Recall our **Ridge Regression** optimisation problem, where we have elevated our input attributes to a *k*-dimensional feature space via the map, $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^k$:

$$\underset{\mathbf{w}}{\text{argmin}} \quad \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{w} \cdot \phi(\mathbf{x}^{(i)}) \right)^2 + \lambda \|\mathbf{w}\|_2^2$$

- Here the so-called **primal solution** is generated by the **revised normal equations**:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{2}$$

  Where **X** is the $n \times k$ feature mapped **design matrix**, $\mathbf{X} = [\phi(\mathbf{x}^{(1)}), \phi(\mathbf{x}^{(2)}), ..., \phi(\mathbf{x}^{(n)})]^T$

- And the associated prediction for a novel test point **z** is:

$$f(\mathbf{z}) = \mathbf{w} \cdot \phi(\mathbf{z}) = \mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\phi(\mathbf{z})$$

Ridge Regression: Dual Form

- Left multiply equation (1) by $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$:

$$\begin{aligned} \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} &= \mathbf{X}^T\mathbf{y} \qquad (3) \\ \mathbf{w} &= \lambda^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{X}^T\alpha \end{aligned}$$

- Here:

$$\begin{aligned} \boldsymbol{\alpha} &= \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ \lambda\alpha &= (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ \lambda\alpha &= (\mathbf{y} - \mathbf{X}\mathbf{X}^T\alpha) \qquad \textbf{From equation (3)} \\ \boldsymbol{\alpha} &= (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y} \end{aligned}$$

This is the so-called **dual solution**

Ridge Regression: Dual Form

- And the associated prediction for a novel test point **z** is:

$$\begin{aligned} f(\mathbf{z}) &= \mathbf{w} \cdot \phi(\mathbf{z}) \\ &= \mathbf{y}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{X}\phi(\mathbf{z}) \\ &= \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}^{(i)}) \cdot \phi(\mathbf{z}) \end{aligned}$$

- In fact the same result follows from:
  - The **Representer Theorem**
  - The **Lagrangian Dual solution**...as we observed earlier

Ridge Regression: Kernel Version

- We may re-write this using kernels as:

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

  Where:
  $\boldsymbol{\alpha} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}$
  $(\mathbf{X}\mathbf{X}^T)_{ij} = \kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$

- We see that $\phi(\mathbf{x})$ is never explicitly evaluated either within the calculation of α_i or within the kernelised prediction function

Ridge Regression: The Value of the Kernel Trick

- We note that the primal solution involves an inversion of the **design matrix**, $\mathbf{X}^T\mathbf{X}$, an operation of complexity $\mathcal{O}(k^3)$
  ...and prediction involves an operation of complexity $\mathcal{O}(k)$

- Meanwhile the dual solution involves an inversion of the **Gram matrix**, $\mathbf{X}\mathbf{X}^T$, an operation of complexity $\mathcal{O}(n^3)$
  ...and prediction involves an operation of complexity $\mathcal{O}(n|\kappa|)$
  (where $|\kappa|$ is the number of operations involved in the caluclation of an inner product)

- Thus, for $k \gg n$ the value of the dual form and the kernel trick becomes apparent

Lecture Overview

Summary

1. We can attempt to enhance the **linear** algorithms of machine learning by affecting a **feature map** of the input attributes in order to elevate them to a higher dimensional **non-linear** space
2. This is a sensible and a tractable strategy if we can employ the **Kernel Trick** and work only via the dot product of feature mapped data points
3. The kernel trick is manifest iff a **Representer Theorem** holds for our optimisation problem
4. In this case, **Mercer's Theorem** offers us a mechanism for defining a valid feature space via **Mercer Kernels**, without the need to ever explicitly define the feature mapping

Lecture Overview

Multiple Constraints: Problem

- $$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x})$$

  $$\text{subject to:} \quad \begin{cases} \{g^{(i)}(\mathbf{x}) \leqslant 0\}_{i=1}^{m} \\ \{h^{(j)}(\mathbf{x}) = 0\}_{j=1}^{p} \end{cases}$$

Multiple Constraints: Lagrangian

- We express the Lagrangian as:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{m} \mu^{(i)} g^{(i)}(\mathbf{x}) + \sum_{j=1}^{p} \lambda^{(j)} h^{(j)}(\mathbf{x})$$

Where:
$\boldsymbol{\lambda} = [\lambda^{(1)}, ..., \lambda^{(p)}]^T, \{\lambda^{(j)} \in \mathbb{R}\}_{j=1}^{p};$
$\boldsymbol{\mu} = [\mu^{(1)}, ..., \mu^{(m)}]^T, \{\mu^{(i)} \in \mathbb{R}^{\geqslant 0}\}_{i=1}^{m};$
are Lagrange multipliers

Multiple Constraints: Problem Reformulation

- And we can solve our problem by seeking stationary solutions $(\mathbf{x}^*, \{\mu^{(i)*}\}, \{\lambda^{(j)*}\})$ which satisfy the following:

$$\nabla_{\mathbf{x}} \mathcal{L} = \mathbf{0}$$

$$\text{subject to:} \quad \begin{cases} \{g^{(i)}(\mathbf{x}) \leqslant 0\}_{i=1}^{m}, \{h^{(j)}(\mathbf{x}) = 0\}_{j=1}^{p} \\ \{\mu^{(i)} \geqslant 0\}_{i=1}^{m} \\ \{\mu^{(i)} g^{(i)}(\mathbf{x}) = 0\}_{i=1}^{m} \end{cases}$$

Duality: Primal Problem

- The original problem is sometimes know as the **primal problem**, and its variables, **x**, are known as the **primal variables**

- It is equivalent to the following formulation:

$$\min_{\mathbf{x}} \left[\max_{\boldsymbol{\lambda},\boldsymbol{\mu}\geqslant 0} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right]$$

- Here the bracketed term is known as the **primal objective** function

Duality: Barrier Function

- We can re-write the primal objective as follows:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\mu}\geqslant 0} \mathcal{L}(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\mu}) = f(\mathbf{x}) + \max_{\boldsymbol{\lambda},\boldsymbol{\mu}\geqslant 0}\left[\sum_{i=1}^{m} \mu^{(i)} g^{(i)}(\mathbf{x}) + \sum_{j=1}^{p} \lambda^{(j)} h^{(j)}(\mathbf{x})\right]$$

- Here the second term gives rise to a **barrier function** which enforces the constraints as follows:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\mu}\geqslant 0}\left[\sum_{i=1}^{m} \mu^{(i)} g^{(i)}(\mathbf{x}) + \sum_{j=1}^{p} \lambda^{(j)} h^{(j)}(\mathbf{x})\right] = \begin{cases} 0 & \text{if } \mathbf{x} \text{ is feasible} \\ \infty & \text{if } \mathbf{x} \text{ is infeasible} \end{cases}$$

Duality: Minimax Inequality

- In order to make use of this barrier function formulation, we will need the **minimax inequality**:

$$\max_{\mathbf{y}} \min_{\mathbf{x}} \phi(\mathbf{x}, \mathbf{y}) \leqslant \min_{\mathbf{x}} \max_{\mathbf{y}} \phi(\mathbf{x}, \mathbf{y})$$

- **Proof:**

$$\min_{\mathbf{x}} \phi(\mathbf{x}, \mathbf{y}) \leqslant \phi(\mathbf{x}, \mathbf{y}) \qquad \forall \mathbf{x}, \mathbf{y}$$

This is true for all **y**, therefore, in particular the following is true:

$$\max_{\mathbf{y}} \min_{\mathbf{x}} \phi(\mathbf{x}, \mathbf{y}) \leqslant \max_{\mathbf{y}} \phi(\mathbf{x}, \mathbf{y}) \qquad \forall \mathbf{x}$$

This is true for all **x**, therefore, in particular the following is true:

$$\max_{\mathbf{y}} \min_{\mathbf{x}} \phi(\mathbf{x}, \mathbf{y}) \leqslant \min_{\mathbf{x}} \max_{\mathbf{y}} \phi(\mathbf{x}, \mathbf{y})$$

Duality: Weak Duality

- We can now introduce the concept of **weak duality**:

$$\min_{\mathbf{x}}\left[\max_{\boldsymbol{\lambda},\boldsymbol{\mu}\geqslant 0}\mathcal{L}(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\mu})\right]\geqslant\max_{\boldsymbol{\lambda},\boldsymbol{\mu}\geqslant 0}\left[\min_{\mathbf{x}}\mathcal{L}(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\mu})\right]$$

- Here the bracketed term on the right hand side is known as the **dual objective** function, $\mathcal{D}(\boldsymbol{\lambda},\boldsymbol{\mu})$
- If we can solve the right hand side of the inequality then we have a lower bound on the solution of our optimisation problem

Duality: Weak Duality

- And often the RHS side of the inequality is an **easier** problem to solve, because:
  - $\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is an **unconstrained** optimisation problem for a given value of $(\boldsymbol{\lambda}, \boldsymbol{\mu})$**...**
  - **...**And if solving this problem is not hard then the overall problem is not hard to solve because:
  - $\max_{\boldsymbol{\lambda}, \boldsymbol{\mu} \geqslant 0} [\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})]$ is a maximisation problem over a set of affine functions - thus it is a **concave maximisation** problem or equivalently a **convex minimisation** problem, and we know that such problems can be efficiently solved
  - Note that this is true regardless of whether $f, g^{(i)}, h^{(j)}$ are nonconvex

Duality: Strong Duality

- For certain classes of problems which satisfy **constraint qualifications** we can go further and **strong duality** holds:

$$\min_{\mathbf{x}} \left[\max_{\boldsymbol{\lambda},\boldsymbol{\mu} \geqslant 0} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right] = \max_{\boldsymbol{\lambda},\boldsymbol{\mu} \geqslant 0} \left[\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right]$$

- There are several different constraint qualifications. One is **Slater's Condition** which holds for **convex optimisation** problems

- Recall, these are problems for which f is convex and $g^{(i)}, h^{(j)}$ are convex sets

- For problems of this type we may seek to solve the **dual optimisation** problem:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\mu} \geqslant 0} \left[\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right]$$

Duality: Strong Duality

- Another reason for adopting the dual optimisation approach to solving contrained optimisation problems is based on dimensionality:
- If the dimensionality of the dual variables, $(m + p)$, is less than the dimensionality of the primal variables, n, then dual optimisation often offers a more efficient route to solutions
- This is of particular importance if we are dealing with infinite dimensional primal variables