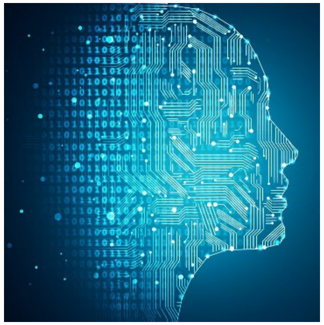
Principles of Machine Learning

Kernels

Kevin Moon (kevin.moon@usu.edu)

STAT/CS 5810/6655



One weird kernel trick

- http://oneweirdkerneltrick.com/

Overview

- So far we have mainly focused on linear methods
- There are several nonlinear methods that we will consider in this course
- The first class of nonlinear methods that we will study are "kernel methods"
- We start here because kernel methods build directly on the linear methods we have just studied

Outline

1. Nonlinear classification via nonlinear feature maps
2. Inner product kernels
3. Symmetric positive definite kernels
4. The kernel trick

Nonlinear feature maps

- One way to create a nonlinear method for regression or classification is to transform the feature vector via a nonlinear feature map
$$\mathbf{\Phi}: \mathbb{R}^d \to \mathbb{R}^m$$
and apply a linear method to the transformed data $\mathbf{\Phi}(\boldsymbol{x}_1), \dots, \mathbf{\Phi}(\boldsymbol{x}_n)$.
- Nonlinear regression:
$$f(\boldsymbol{x}) = \boldsymbol{w}^T \Phi(\boldsymbol{x}) + b$$
  - $\boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}$
- Nonlinear classification:
$$f(\boldsymbol{x}) = \operatorname{sign}\{\boldsymbol{w}^T \Phi(\boldsymbol{x}) + b\}$$
  - $\boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}$

Polynomial Regression

- Determine the least squares cubic polynomial fit to training data $(x_1, y_1), \dots, (x_n, y_n)$ where $x_i, y_i \in \mathbb{R}$

- Can write
$$f(x) = a + bx + cx^2 + dx^3 = \boldsymbol{w}^T \boldsymbol{\Phi}(x) + a$$
where
$$w = \begin{bmatrix} b \\ c \\ d \end{bmatrix}, \qquad \boldsymbol{\Phi}(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

- Note that $m = 3$ and $\boldsymbol{\Phi}(x)$ is a nonlinear function

Polynomial Regression

- The empirical risk (using squared error loss):

$$\sum_{i=1}^{n}\left(y_i - f(x_i)\right)^2 = \sum_{i=1}^{n}\left(y_i - \boldsymbol{w}^T\boldsymbol{\Phi}(x) - a\right)^2$$

- The minimizer is:

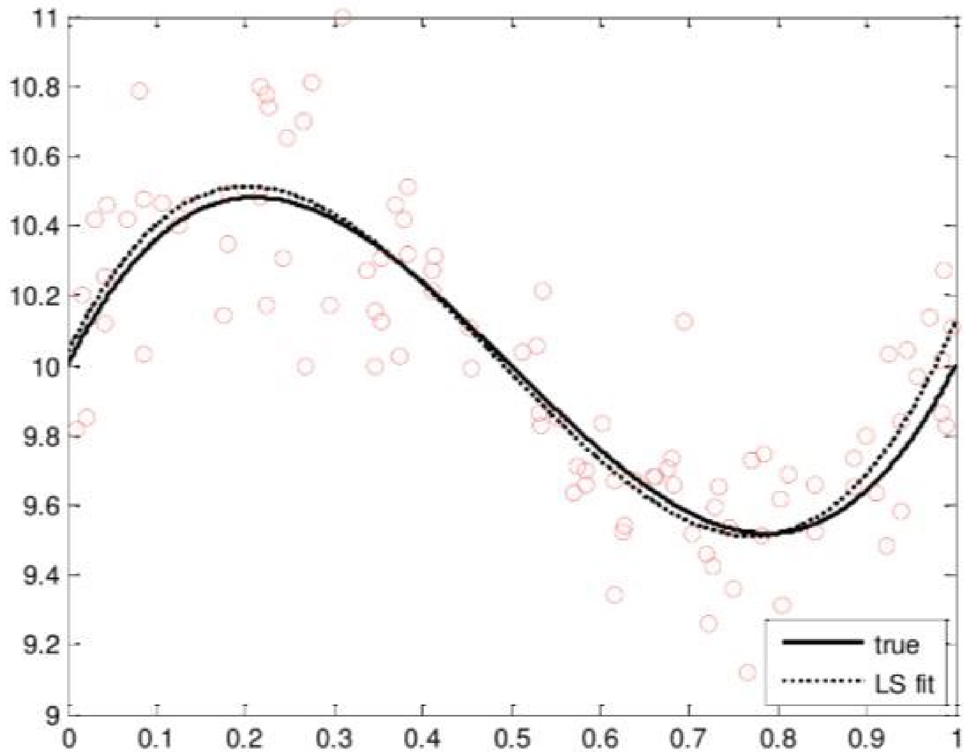
$$\begin{bmatrix} a \\ \boldsymbol{w} \end{bmatrix} = \left(X^T X\right)^{-1} X^T \boldsymbol{y}$$

where

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \qquad X = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

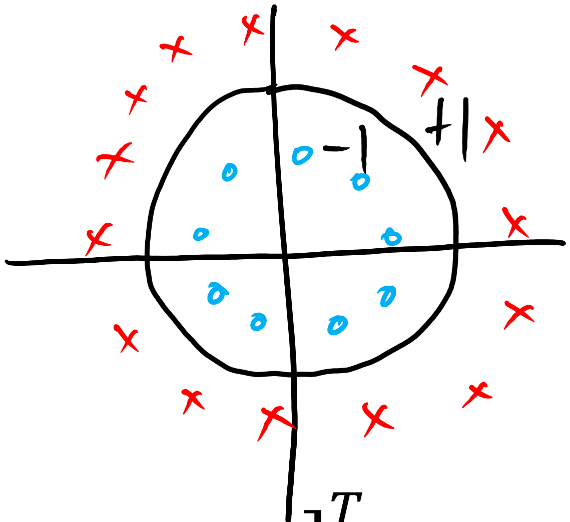
Polynomial Regression

- For large m (i.e. polynomial degree) relative to n, regularization becomes necessary
- Otherwise, the matrix $X^T X$ becomes ill-conditioned and regularization is needed to avoid overfitting



Binary Classification

- Write $\boldsymbol{x} = [x^{(1)} \quad x^{(2)}]^T \in \mathbb{R}^2$



- Consider

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \left[\left(x^{(1)}\right)^2 \quad \left(x^{(2)}\right)^2 \quad x^{(1)} \quad x^{(2)}\right]^T$$

Binary Classification

- Training data are separated by a circular classifier
$$\boldsymbol{x} \mapsto \operatorname{sign}\left\{\left(x^{(1)} - c^{(1)}\right)^2 + \left(x^{(2)} - c^{(2)}\right)^2 - r^2\right\}$$
for a certain radius r and center
$$\boldsymbol{c} = \begin{bmatrix} c^{(1)} \\ c^{(2)} \end{bmatrix}$$
- This is a linear classifier in the transformed space where
$$\boldsymbol{w} = \begin{bmatrix} 1 \\ 1 \\ -2c^{(1)} \\ -2c^{(2)} \end{bmatrix} \quad b = \left(c^{(1)}\right)^2 + \left(c^{(2)}\right)^2 - r^2$$
- In this example, you can also absorb the offset b into $\boldsymbol{\Phi}$

Inner Product Kernels

- Problem with previous approach: m can explode as d increases
  - Thus it can be difficult to store/compute/manipulate $\mathbf{\Phi}$ directly
- Fortunately, the following facts allow us to use nonlinear feature maps for large d:
  - Many ML algorithms depend on $\mathbf{\Phi}(\boldsymbol{x})$ only via inner products
$$\langle \mathbf{\Phi}(\boldsymbol{x}), \mathbf{\Phi}(\boldsymbol{x}') \rangle.$$
  - For certain $\mathbf{\Phi}$, the function
$$k(x, x') \coloneqq \langle \mathbf{\Phi}(\boldsymbol{x}), \mathbf{\Phi}(\boldsymbol{x}') \rangle$$
    can be computed efficiently even if m is huge or infinite!
- k is called an *inner product kernel*
- Let's look at some examples using the *dot product*

Homogeneous Polynomial Kernel

- Degree $p = 2, d = 2$

$$
\begin{aligned}
k(\boldsymbol{u}, \boldsymbol{v}) &= \left(\boldsymbol{u}^T \boldsymbol{v}\right)^2 \\
&= \left(u^{(1)} v^{(1)} + u^{(2)} v^{(2)}\right)^2 \\
&= \left(u^{(1)}\right)^2 \left(v^{(1)}\right)^2 + 2u^{(1)} u^{(2)} v^{(1)} v^{(2)} + \left(u^{(2)}\right)^2 \left(v^{(2)}\right)^2 \\
&= \langle \boldsymbol{\Phi}(\boldsymbol{u}), \boldsymbol{\Phi}(\boldsymbol{v}) \rangle
\end{aligned}
$$

where

$$
\boldsymbol{\Phi}(\boldsymbol{u}) = \left[\left(u^{(1)}\right)^2, \sqrt{2} u^{(1)} u^{(2)}, \left(u^{(2)}\right)^2\right]^{\mathrm{T}}.
$$

Homogeneous Polynomial Kernel

- Degree $p = 2, d$ arbitrary

$$
\begin{aligned}
k(\boldsymbol{u}, \boldsymbol{v}) &= \left(\boldsymbol{u}^T \boldsymbol{v}\right)^2 \\
&= \left(\sum_{i=1}^{d} u^{(i)} v^{(i)}\right)^2 \\
&= \sum_{i=1}^{d} \left(u^{(i)}\right)^2 \left(v^{(i)}\right)^2 + \sum_{i<j} 2u^{(i)} u^{(j)} v^{(i)} v^{(j)} \\
&= \langle \boldsymbol{\Phi}(\boldsymbol{u}), \boldsymbol{\Phi}(\boldsymbol{v}) \rangle
\end{aligned}
$$

where

$$
\boldsymbol{\Phi}(\boldsymbol{u}) = \left[\left(u^{(1)}\right)^2, \ldots, \left(u^{(d)}\right)^2, \sqrt{2} u^{(1)} u^{(2)}, \ldots, \sqrt{2} u^{(d-1)} u^{(d)}\right]^{\mathrm{T}}.
$$

- $m = d + \frac{d(d-1)}{2}$

Group Exercise

1. Let $k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^T \boldsymbol{v})^3$ where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^2$. Find $\boldsymbol{\Phi}$ such that

$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{\Phi}(u), \boldsymbol{\Phi}(v) \rangle$$

2. The *inhomogeneous polynomial kernel of degree* 2 is

$$k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^T \boldsymbol{v} + 1)^2$$

   where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^2$. Determine a feature map $\boldsymbol{\Phi}$ such that $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{\Phi}(\boldsymbol{u}), \boldsymbol{\Phi}(\boldsymbol{v}) \rangle$.

Remarks

- General homogoneous polynomial kernel:

$$k(\boldsymbol{u},\boldsymbol{v}) = (\boldsymbol{u}^T\boldsymbol{v})^p$$

$$= \sum_{(j_1,\ldots,j_d):\sum j_i = p} \binom{p}{j_1 \ldots j_d} (u^{(1)})^{j_1} \cdots (u^{(d)})^{j_d} (v^{(1)})^{j_1} \cdots (v^{(d)})^{j_d}$$

$$\Rightarrow \boldsymbol{\Phi}(\boldsymbol{u}) = \left[\ldots, \sqrt{\binom{p}{j_1 \ldots j_d}} (u^{(1)})^{j_1} \ldots (u^{(1)})^{j_d}, \ldots\right]^T$$

- For the general inhomogeneous polynomial kernel $\boldsymbol{\Phi} =$ all monomials in d variables up to degree p

- All the preceding examples involve the dot product, but some important kernels involve other inner products

Inner Products

- A *(real) inner product space* is a vector space V on which we can define a function $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ (called an *inner product*) such that

  1. $\forall \alpha_1, \alpha_2 \in \mathbb{R}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{v} \in V$
  $$\langle \alpha_1 \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2, \boldsymbol{v} \rangle = \alpha_1 \langle \boldsymbol{u}_1, \boldsymbol{v} \rangle + \alpha_2 \langle \boldsymbol{u}_2, \boldsymbol{v} \rangle$$
  2. $\forall\ \boldsymbol{u}, \boldsymbol{v} \in V$
  $$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \langle \boldsymbol{v}, \boldsymbol{u} \rangle$$
  3. $\forall \boldsymbol{u} \in V$,
  $$\langle \boldsymbol{u}, \boldsymbol{u} \rangle \geq 0,$$
  with equality iff $\boldsymbol{u} = \boldsymbol{0}$.

- Recall: We say $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is an *inner product kernel* if $\exists$ an inner product space V and a feature map $\boldsymbol{\Phi} : \mathbb{R}^d \to V$ such that
$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{\Phi}(\boldsymbol{u}), \boldsymbol{\Phi}(\boldsymbol{v}) \rangle \quad \forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$$

- *Note*: $\boldsymbol{\Phi}$ and V are not unique for a given k

Symmetric Positive Definite Kernels

- One way to determine an IP Kernel is to construct Φ explicitly as we did in the examples above.
- We can also verify that k is an IP kernel if it satisfies the following properties.
- Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. We say k is *symmetric* if $k(\boldsymbol{u}, \boldsymbol{v}) = k(\boldsymbol{v}, \boldsymbol{u})\ \forall \boldsymbol{u}, \boldsymbol{v}$.
- We say k is *positive definite* if

$$\begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}$$

  is a positive *semi*-definite matrix for all $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$
- If k is both symmetric and positive definite, it is referred to as a *symmetric, positive definite (SPD) kernel.*
- **Theorem:** k is an SPD kernel $\iff$ k is an inner product kernel

Important Kernels

- Homogeneous polynomial kernel

$$k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^T \boldsymbol{v})^p$$

- Inhomogeneous polynomial kernel

$$k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^T \boldsymbol{v} + c)^p, c > 0$$

- Gaussian kernel

$$k(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\frac{1}{2\sigma^2} \|\boldsymbol{u} - \boldsymbol{v}\|^2\right), \sigma > 0$$

- For the Gaussian kernel, V is infinite dimensional!
- This is ok, though. In kernel methods we don't have to work with $\boldsymbol{\Phi}$, just k

Big Picture: The Kernel Trick

- Using kernels, we can obtain nonlinear methods from linear methods as follows:
  1. Select an IP/SPD kernel k
  2. Formulate your linear method such that feature vectors (i.e., the training data and an arbitrary test instance) only appear via inner products $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$
  3. Replace $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ with
  $$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}') \rangle$$
  throughout the algorithm
- This idea is called the *kernel trick*
- The resulting method is equivalent to applying the original linear method to the non-linearly transformed data $(\boldsymbol{\Phi}(x_1), y_1), \dots, (\boldsymbol{\Phi}(x_n), y_n)$. With the kernel trick, we never have to compute $\boldsymbol{\Phi}$, just k.

Applications of the kernel trick

- Many standard methods can be kernelized
  - Ridge regression (kernel ridge regression)
    - Learns a nonlinear regression function
  - PCA (kernel PCA)
    - Learns a nonlinear dimensionality reduction
  - Optimal soft-margin hyperplane (the support vector machine)
    - Learns a nonlinear decision boundary
  - Others…
- We'll cover kernel ridge regression and the support vector machine in the next few lectures

Reproducing Kernel Hilbert Spaces (RKHS)

- Can define a space (set) of real-valued functions defined in terms of a positive definite kernel.
  - Referred to as an RKHS
- Optimizing over an RKHS can give many ML algorithms
  - Kernel ridge regression
  - SVM
  - Kernel logistic regression
  - Semi-supervised learning
- Provides a very rich theory that is often used to develop new methods
- If time, we'll go into more detail later in the course

Further reading

- ESL Section 5.8