Kernel Methods

Announcements

- HW3 due today, HW4 assigned

Outline

- Nonlinear classification and regression via nonlinear feature maps
- Kernels
- Kernel Methods

Nonlinear Feature Maps

- One way to create a nonlinear method for regression or classification is to first transform the feature vector via a *nonlinear feature map*

$$x \longmapsto \Phi(x), \quad \Phi: \mathbb{R}^d \to \mathbb{R}^m$$

and apply a linear method to the transformed data

- In regression, the nonlinear function model is

$$f(x) = w^T \Phi(x) + b$$

where $\boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}$

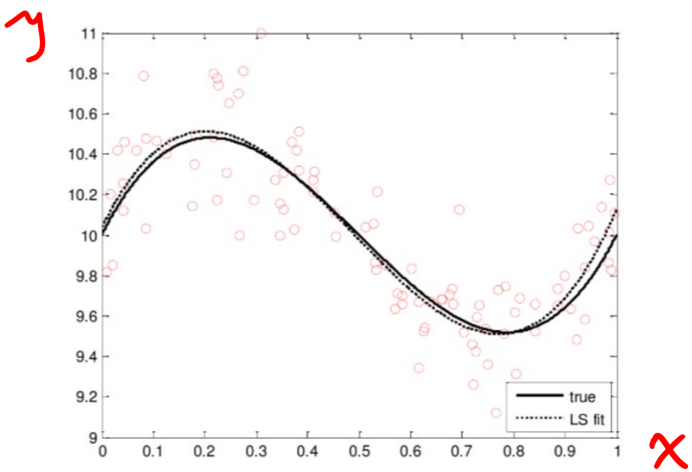
- In binary classification, the nonlinear classifier model is

$$h(x) = \text{sign}\left(w^T \Phi(x) + b\right)$$

where $\boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}$

Nonlinear Feature Maps

- **Example:** cubic polynomial regression



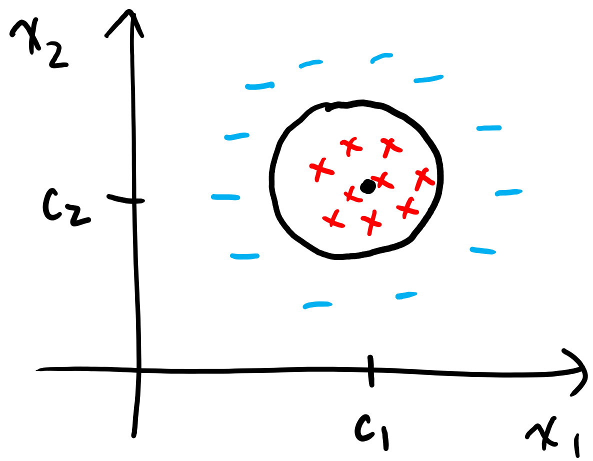
$$\Phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}, \quad w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$\min_{w,b} \frac{1}{n}\sum_{i=1}^{n}\Big(y_i - \big[\underbrace{w_3 x_i^3 + w_2 x_i^2 + w_1 x_i}_{w^T \Phi(x_i)} + b\big]\Big)^2$$

Nonlinear Feature Maps

- **Example:** circular classifier

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



$$x \mapsto \text{sign}\left\{ (x_1 - c_1)^2 + (x_2 - c_2)^2 - r^2 \right\}$$

$$= \text{sign}\left\{ x_1^2 - 2c_1 x_1 + c_1^2 + x_2^2 - 2c_2 x_2 + c_2^2 - r^2 \right\}$$

$$= \text{sign}\left\{ w^T \Phi(x) + b \right\}$$

$$\Phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}, \quad w = \begin{bmatrix} 1 \\ 1 \\ -2c_1 \\ -2c_2 \\ 0 \end{bmatrix}, \quad b = c_1^2 + c_2^2 - r^2$$

Inner Product Kernels

- Problem with the above approach: m can explode as d increases
- This makes it prohibitive to compute/store/manipulate $\mathbf{\Phi}$ directly
- Fortunately, the following facts allow us to use nonlinear feature maps for large m:
  - Many ML algorithms depend on $\mathbf{\Phi}(\boldsymbol{x})$ only via *inner products*

    $$\langle \Phi(x), \Phi(x') \rangle$$

  - For certain $\mathbf{\Phi}$, the function

    $$k(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

    can be computed efficiently even if m is huge or possibly infinite!
- k is called an inner product kernel
- Let's look at an example involving the *dot product*

Example

- 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 $\mathbf{\Phi}$ such that

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

$$\begin{aligned} k(u,v) &= (u_1 v_1 + u_2 v_2 + 1)^2 \\ &= u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2 + 2 u_1 v_1 \\ &\quad + 2 u_2 v_2 + 1 \\ &= \langle \Phi(u), \Phi(v) \rangle, \quad \Phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ \sqrt{2} u_1 u_2 \\ \sqrt{2} u_1 \\ \sqrt{2} u_2 \\ 1 \end{bmatrix} \end{aligned}$$

Important Kernels

- Homogeneous polynomial kernel

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

- Inhomogeneous polynomial

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

- 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, $m = \infty$ and the dot product is replaced by the ℓ_2 inner product: $\langle (a_i), (b_i) \rangle = \sum_{i=1}^{\infty} a_i b_i$.

SPD 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 $n \in \mathbb{N}$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$
- 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

$\iff \exists\, \Phi$ s.t. $k(u,v) = \langle \Phi(u), \Phi(v) \rangle$

The Kernel Trick

- A machine learning algorithm is said to be *kernelizable* if it is possible to formulate the algorithm such that all training instances $\boldsymbol{x}_i$ and any test instance $\boldsymbol{x}$ occur exclusively in inner products of the form $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$, $\langle \boldsymbol{x}_i, \boldsymbol{x} \rangle$ or $\langle \boldsymbol{x}, \boldsymbol{x} \rangle$.

- Suppose $\boldsymbol{\Phi}$ is a feature map associated to an inner product kernel k.

- If we apply a kernelizable algorithm to the training data

$$(\Phi(x_1), y_1), \ldots, (\Phi(x_n), y_n)$$

then we can formulate the algorithm such that transformed feature vectors only appear via inner products $\langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}') \rangle$ with other transformed feature vectors.

- Can implement by evaluating $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$

which eliminates the need to ever compute $\boldsymbol{\Phi}(\boldsymbol{x})$ explicitly.

Example: Nearest Neighbors

- Given training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, the 1-nearest neighbor classifier assigns a test point $\boldsymbol{x}$ to the label

$$y_{\operatorname{argmin}_{1 \le i \le n} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2}$$

- Suppose Φ is a feature map associated to an inner product/SPD kernel k, i.e.,

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

  for all $\boldsymbol{u}, \boldsymbol{v}$.

- Now consider the 1-NN classifier after first applying Φ:

$$y_{\operatorname{argmin}_{1 \le i \le n} \|\Phi(x) - \Phi(x_i)\|_2}$$

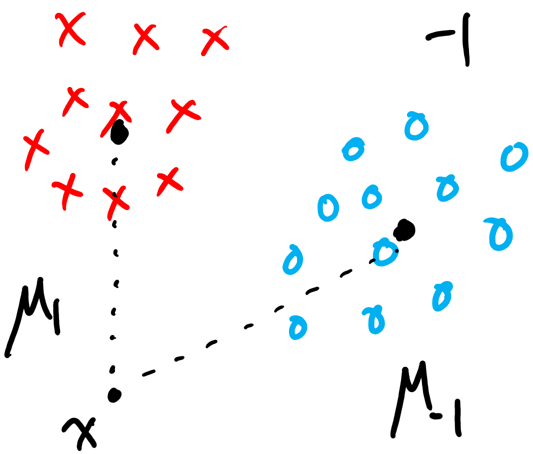
$$\|\Phi(x) - \Phi(x_i)\|_2^2 = (\Phi(x) - \Phi(x_i))^T (\Phi(x) - \Phi(x_i))$$

$$= \langle \Phi(x), \Phi(x) \rangle - 2 \langle \Phi(x), \Phi(x_i) \rangle + \langle \Phi(x_i), \Phi(x_i) \rangle$$

$$= k(x, x) - 2k(x, x_i) + k(x_i, x_i)$$

+1

Nearest Centroid Classifier



$$f(x) = \text{sign}\left\{ \|x - \mu_{-1}\|^2 - \|x - \mu_1\|^2 \right\}$$

$$\mu_1 = \frac{1}{n_+} \sum_{i:y_i=1} x_i$$

$$\mu_{-1} = \frac{1}{n_-} \sum_{i:y_i=-1} x_i$$

$$f(\boldsymbol{x}) = \text{sign}\left\{ \frac{1}{n_+} \sum_{i:y_i=1} \overbrace{\langle \boldsymbol{x}_i, \boldsymbol{x} \rangle}^{k(x_i,x)} - \frac{1}{n_-} \sum_{i:y_i=-1} \overbrace{\langle \boldsymbol{x}_i, \boldsymbol{x} \rangle}^{k(x_i,x)} + \frac{1}{2} \left(\frac{1}{n_+^2} \sum_{\substack{i:y_i=1 \\ j:y_j=1}} \overbrace{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle}^{k(x_i,x_j)} - \frac{1}{n_-^2} \sum_{\substack{i:y_i=-1 \\ j:y_j=-1}} \overbrace{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle}^{k(x_i,x_j)} \right) \right\}$$

Poll

True or false: To apply the kernel trick using a kernel k, we need to know the output dimension of the feature map Φ associated to k.

(A) True

(B) False

Kernel Ridge Regression

- We will argue that ridge regression is kernelizable, and use the kernel trick to extend it to a nonlinear regression method called *kernel ridge regression.*
- To simplify the presentation, we'll consider ridge regression *without offset*

$$\min_{w} \frac{1}{n} \sum (y_i - w^T x_i)^2 + \lambda \|w\|^2$$

- Since some $\mathbf{\Phi}$ contain a constant term, as with the inhomogeous polynomial kernel, the offset is not always needed.
- KRR with offset is described in my lecture notes.

Poll

- Introduce the notation

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$$

- The solution to

$$\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 + \lambda \|\boldsymbol{w}\|^2$$

is

(A) $\widehat{\boldsymbol{w}} = (\boldsymbol{X}^T \boldsymbol{X} + n\lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y}$

(B) $\widehat{\boldsymbol{w}} = (\boldsymbol{X} \boldsymbol{X}^T + n\lambda \boldsymbol{I})^{-1} \boldsymbol{X} \boldsymbol{y}$

(C) $\widehat{\boldsymbol{w}} = (\boldsymbol{X} \boldsymbol{X}^T + n\lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y}$

(D) $\widehat{\boldsymbol{w}} = (\boldsymbol{X}^T \boldsymbol{X} + n\lambda \boldsymbol{I})^{-1} \boldsymbol{X} \boldsymbol{y}$

Kernel Ridge Regression

- Note $\boldsymbol{X}^T\boldsymbol{X}$ is *not* the Gram matrix of the training data.
- Idea: apply matrix inversion lemma:

$$(\mathbf{P}+\mathbf{QRS})^{-1} = \mathbf{P}^{-1} - \mathbf{P}^{-1}\mathbf{Q}(\mathbf{R}^{-1}+\mathbf{SP}^{-1}\mathbf{Q})^{-1}\mathbf{SP}^{-1}$$

- After simplification, the solution of RR without offset is

$$\begin{aligned}
\widehat{f}(\boldsymbol{x}) &= \widehat{\boldsymbol{w}}^T\boldsymbol{x} \\
&= \boldsymbol{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X}+n\lambda\boldsymbol{I})^{-1}\boldsymbol{x} \\
&= \boldsymbol{y}^T(\boldsymbol{X}\boldsymbol{X}^T+n\lambda\boldsymbol{I})^{-1}\underbrace{\boldsymbol{X}}\boldsymbol{x} \\
&= \boldsymbol{y}^T(\boldsymbol{K}+n\lambda\boldsymbol{I})^{-1}\boldsymbol{k}(\boldsymbol{x})
\end{aligned}$$

MIL + algebra

where

$$\boldsymbol{K} = \begin{bmatrix} \langle\boldsymbol{x}_1,\boldsymbol{x}_1\rangle & \cdots & \langle\boldsymbol{x}_1,\boldsymbol{x}_n\rangle \\ \vdots & \ddots & \vdots \\ \langle\boldsymbol{x}_n,\boldsymbol{x}_1\rangle & \cdots & \langle\boldsymbol{x}_n,\boldsymbol{x}_n\rangle \end{bmatrix} \qquad \boldsymbol{k}(\boldsymbol{x}) = \begin{bmatrix} \langle\boldsymbol{x}_1,\boldsymbol{x}\rangle \\ \vdots \\ \langle\boldsymbol{x}_n,\boldsymbol{x}\rangle \end{bmatrix}$$

Kernel Ridge Regression

- If we redefine

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

then the function estimated by KRR is

$$\hat{f}(x) = \underbrace{y^T (K + n\lambda I)^{-1}}_{\alpha^T} k(x)$$

- The form of the estimated function is

$$\hat{f}(x) = \alpha^T k(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

Summary

- Kernels: Convenient way to incorporate nonlinear feature maps into an algorithm
- Kernel ridge regression: powerful nonlinear regression method, useful element of the ML toolbox
- See lecture notes for KRR with offset
- Neural tangent kernel: interesting connections to neural networks
- Next two lectures: We will kernelize the optimal soft-margin hyperplane classifier, leading to the support vector machine

$$k(x,x') = \exp\left(-\gamma \|x-x'\|^2\right)$$

$d=1$

$$\sum \alpha_i k(x,x_i)$$