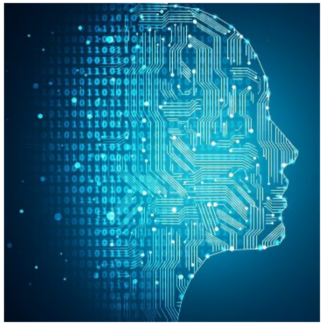
Machine Learning

Kernel Ridge Regression

Kevin Moon (kevin.moon@usu.edu)

STAT/CS 5810/6655



Outline

1. Review of kernels
2. Kernel ridge regression
3. Kernel nearest centroid classifier
4. Kernel ridge regression revisited

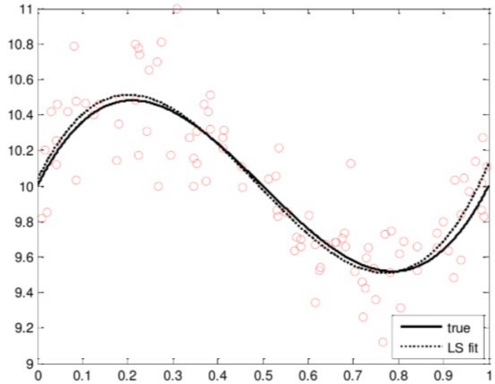
Nonlinear Feature Maps

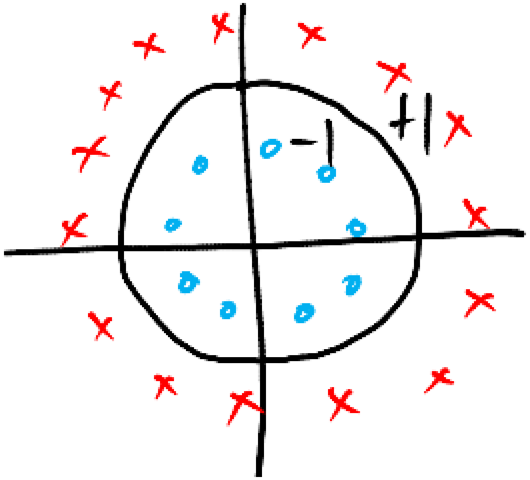
- One way to create a nonlinear method for regression or classification is to first transform the feature vector via a *nonlinear feature map*

$$\mathbf{\Phi} : \mathbb{R}^d \to \mathbb{R}^m$$

and apply a linear method to the transformed data $\mathbf{\Phi}(\boldsymbol{x}_1), ..., \mathbf{\Phi}(\boldsymbol{x}_n)$.

- **Examples:** polynomial regression function, circular classifier
- Problem with the above approach: m can explode as d increases
- This makes it prohibitive to compute/store/manipulate $\mathbf{\Phi}$ directly
- This is where *inner product kernels* save the day





Inner Product Kernels

- 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 $\mathbf{\Phi} : \mathbb{R}^d \to V$ such that

$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \mathbf{\Phi}(\boldsymbol{u}), \mathbf{\Phi}(\boldsymbol{v}) \rangle \quad \forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$$

- Homogeneous polynomial kernel

$$\begin{aligned} k(\boldsymbol{u}, \boldsymbol{v}) &= (\boldsymbol{u}^T \boldsymbol{v})^p \\ \mathbf{\Phi}(\boldsymbol{u}) &= \text{ monomials in } u^{(1)}, \ldots, u^{(d)} \text{ of degree } p \end{aligned}$$

- Inhomogeneous polynomial

$$\begin{aligned} k(\boldsymbol{u}, \boldsymbol{v}) &= (\boldsymbol{u}^T \boldsymbol{v} + c)^p, \ c > 0 \\ \mathbf{\Phi}(\boldsymbol{u}) &= \text{ monomials in } u^{(1)}, \ldots, u^{(d)} \textit{ up to } \text{degree } p \end{aligned}$$

- Gaussian kernel

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

The Kernel Trick

- A machine learning algorithm is said to be *kernelizable* if it can be formulated such that feature vectors (i.e., the training data and an arbitrary test instance) only appear via inner products $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ with other feature vectors.
- Suppose $\boldsymbol{\Phi}$ is a feature map associated with an inner product kernel k.
- If we apply a kernelizable algorithm to the training data $(\boldsymbol{\Phi}(\boldsymbol{x}_1), y_1), \ldots, (\boldsymbol{\Phi}(\boldsymbol{x}_n), y_n)$, then we can formulate the algorithm such that transformed features only appear via inner products $\langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}') \rangle$ with other transformed features
- Can implement by evaluating $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}') \rangle$ which eliminates the need to ever compute $\boldsymbol{\Phi}(\boldsymbol{x})$ explicitly
- In practice, we don't even need to know $\boldsymbol{\Phi}$, we just need to specify an inner product kernel k

Kernel Ridge Regression (1)

- Ridge regression solves

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

- We will show 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 first consider ridge regression without offset

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

- Since some $\boldsymbol{\Phi}$ contain a constant term, as with the inhomogeneous polynomial, an offset isn't always needed

Kernel Ridge Regression (2)

- We can rewrite the objective function as follows (without changing the solution):

$$\|\boldsymbol{y} - X\boldsymbol{w}\|^2 + n\lambda\|\boldsymbol{w}\|^2$$

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \qquad X = \begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(d)} \\ \vdots & & \vdots \\ x_n^{(1)} & \cdots & x_n^{(d)} \end{bmatrix}$$

Kernel Ridge Regression (3)

- The solution is

$$\hat{\boldsymbol{w}} = \left(X^T X + n\lambda I\right)^{-1} X^T \boldsymbol{y}$$

- Problem: $\left[X^T X\right]_{ij} \neq \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$
  - I.e. $X^T X$ is <u>not</u> the gram matrix of the training data
- Idea: apply the matrix inversion lemma:

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

with

$$P = n\lambda I = \mu I, \qquad Q = X^T,$$
$$S = X, \qquad R = I$$

Kernel Ridge Regression (4)

- Plugging in:

$$(\mu I + X^T X)^{-1} = \frac{1}{\mu} I - \frac{1}{\mu} I X^T \left(I + \frac{1}{\mu} X X^T \right)^{-1} X \frac{1}{\mu} I$$
$$= \frac{1}{\mu}\left(I - X^T (\mu I + X X^T)^{-1} X \right)$$

- Therefore,

$$\widehat{\boldsymbol{w}} = (X^T X + n\lambda I)^{-1} X^T \boldsymbol{y} = \frac{1}{\mu}\left(X^T - X^T (\mu I + X X^T)^{-1} X X^T \right) \boldsymbol{y}$$
$$= \frac{1}{\mu}\left(X^T - X^T (G + \mu I)^{-1} G \right) \boldsymbol{y}$$

where

$$G = \begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{x}_1 \rangle & \dots & \langle \boldsymbol{x}_1, \boldsymbol{x}_n \rangle \\ \vdots & & \vdots \\ \langle \boldsymbol{x}_n, \boldsymbol{x}_1 \rangle & \dots & \langle \boldsymbol{x}_n, \boldsymbol{x}_n \rangle \end{bmatrix}$$

Kernel Ridge Regression (5)

- The ridge regression function is therefore

$$
\begin{aligned}
\hat{f}(\boldsymbol{x}) &= \boldsymbol{w}^T \boldsymbol{x} \\
&= \frac{1}{\mu} \boldsymbol{y}^T \left(X - G(G + \mu I)^{-1} X \right) \boldsymbol{x} \\
&= \frac{1}{\mu} \boldsymbol{y}^T \left(I - G(G + \mu I)^{-1} \right) X \boldsymbol{x}
\end{aligned}
$$

Kernel Ridge Regression (6)

- Simplifying $\hat{f}(\boldsymbol{x}) = \frac{1}{\mu}\boldsymbol{y}^T\left(I - G(G+\mu I)^{-1}\right)X\boldsymbol{x}$

$$\begin{aligned} I - G(G+\mu I)^{-1} &= (G+\mu I)(G+\mu I)^{-1} - G(G+\mu I)^{-1} \\ &= (G+\mu I - G)(G+\mu I)^{-1} \\ &= \mu(G+\mu I)^{-1} \end{aligned}$$

$$X\boldsymbol{x} = \begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{x} \rangle \\ \vdots \\ \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle \end{bmatrix}$$

Kernel Ridge Regression (7)

- In summary, we have expressed the solution of ridge regression w/o offset as

$$\hat{f}(\boldsymbol{x}) = \boldsymbol{y}^T(G + n\lambda I)^{-1}\boldsymbol{g}(\boldsymbol{x})$$

where

$$XX^T = G = \begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{x}_1 \rangle & \dots & \langle \boldsymbol{x}_1, \boldsymbol{x}_n \rangle \\ \vdots & & \vdots \\ \langle \boldsymbol{x}_n, \boldsymbol{x}_1 \rangle & \dots & \langle \boldsymbol{x}_n, \boldsymbol{x}_n \rangle \end{bmatrix}$$

$$\boldsymbol{g}(\boldsymbol{x}) = X\boldsymbol{x} = \begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{x} \rangle \\ \vdots \\ \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle \end{bmatrix}$$

Kernel Ridge Regression (8)

- Now suppose we first transform the feature vectors by a nonlinear feature map $\mathbf{\Phi}$, and then apply ridge regression (w/o offset) in the new feature space.
- Our regression function estimate is then

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

where

$$K = \begin{bmatrix} \langle \mathbf{\Phi}(\boldsymbol{x}_1), \mathbf{\Phi}(\boldsymbol{x}_1) \rangle & \dots & \langle \mathbf{\Phi}(\boldsymbol{x}_1), \mathbf{\Phi}(\boldsymbol{x}_n) \rangle \\ \vdots & & \vdots \\ \langle \mathbf{\Phi}(\boldsymbol{x}_n), \Phi(\boldsymbol{x}_1) \rangle & \dots & \langle \mathbf{\Phi}(\boldsymbol{x}_n), \mathbf{\Phi}(\boldsymbol{x}_n) \rangle \end{bmatrix}$$

$$\boldsymbol{k}(\boldsymbol{x}) = \begin{bmatrix} \langle \mathbf{\Phi}(\boldsymbol{x}_1), \mathbf{\Phi}(\boldsymbol{x}) \rangle \\ \vdots \\ \langle \mathbf{\Phi}(\boldsymbol{x}_n), \mathbf{\Phi}(\boldsymbol{x}) \rangle \end{bmatrix}$$

Kernel Ridge Regression (9)

- If $\mathbf{\Phi}$ is the feature map associated with an inner product kernel k, then

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

- Conclusion: we don't need to compute $\mathbf{\Phi}(\boldsymbol{x})$

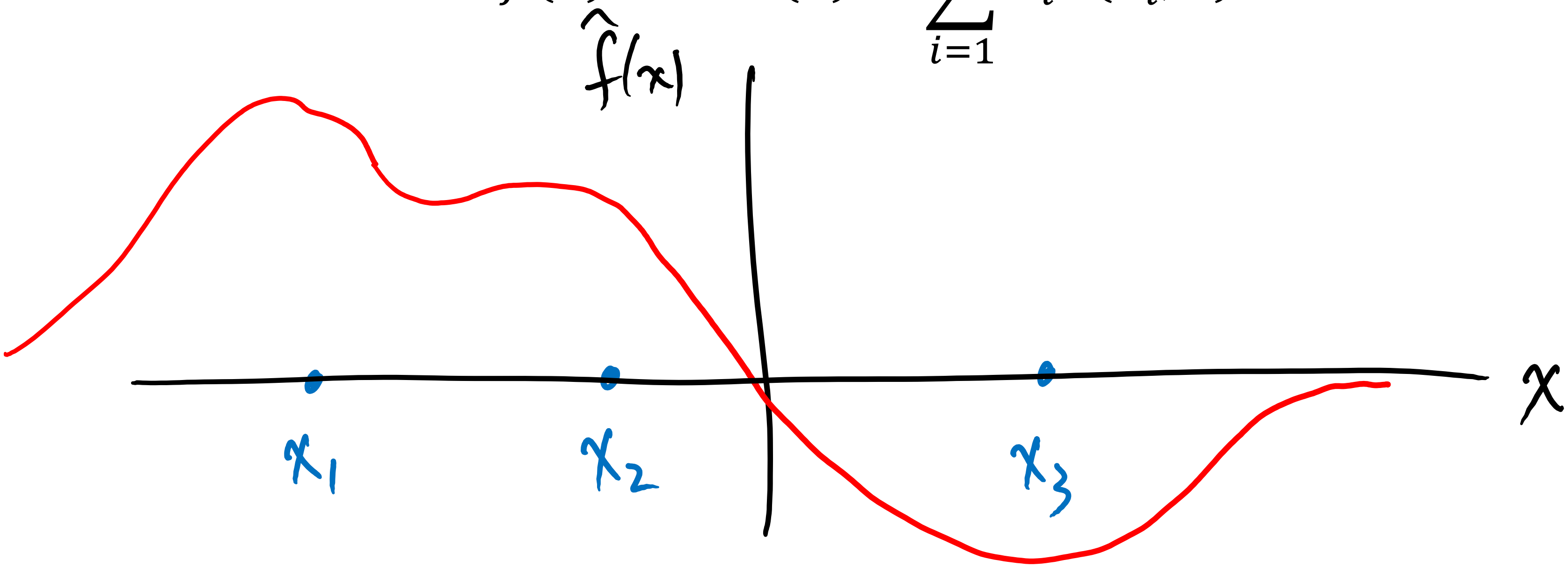
KRR (w/o offset) + Gaussian Kernel

- Consider the Gaussian kernel

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

- The KRR (w/o offset) predictor can be expressed as

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



- Although the Φ associated with the Gaussian kernel does not contain a constant, KRR with the Gaussian kernel often performs well w/o the offset

Group Exercise

Nearest centroid classifer: Given training data $(\boldsymbol{x}_i, y_i), i = 1, \ldots, n$, $y_i \in \{-1, +1\}$, define the *centroids*

$$\boldsymbol{m}_- = \frac{1}{n_-} \sum_{i:y_i=-1} \boldsymbol{x}_i \qquad \boldsymbol{m}_+ = \frac{1}{n_+} \sum_{i:y_i=+1} \boldsymbol{x}_i$$

where $n_- = |\{i : y_i = -1\}|$ and $n_+ = |\{i : y_i = +1\}|$. The nearest centroid classifier is

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sign}\{\|\boldsymbol{x} - \boldsymbol{m}_-\|^2 - \|\boldsymbol{x} - \boldsymbol{m}_+\|^2\} \\ &= \operatorname{sign}\left\{\left(\boldsymbol{x} - \frac{\boldsymbol{m}_+ + \boldsymbol{m}_-}{2}\right)^T (\boldsymbol{m}_+ - \boldsymbol{m}_-)\right\} \end{aligned}$$

1. Kernelize the nearest centroid classifier.

KRR with Offset

- Now assume there is an offet:

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

- Recall the solution is

$$\widehat{\boldsymbol{w}} = (\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}} + n\lambda\mathbf{I})^{-1}\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}$$
$$\widehat{b} = \bar{y} - \widehat{\boldsymbol{w}}^T\bar{\boldsymbol{x}}$$

where

$$\tilde{\boldsymbol{y}} = \begin{bmatrix}\tilde{y}_1\\ \vdots\\ \tilde{y}_n\end{bmatrix}, \quad \tilde{\boldsymbol{X}} = \begin{bmatrix}\tilde{\boldsymbol{x}}_1^T\\ \vdots\\ \tilde{\boldsymbol{x}}_n^T\end{bmatrix}, \quad \begin{aligned}\tilde{\boldsymbol{x}}_i &= \boldsymbol{x}_i - \bar{\boldsymbol{x}} \in \mathbb{R}^d\\ \tilde{y}_i &= y_i - \bar{y} \in \mathbb{R}\\ \bar{\boldsymbol{x}} &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\\ \bar{y} &= \frac{1}{n}\sum_{i=1}^{n}y_i\end{aligned}$$

KRR with Offset

- Using the exact same algebra as before, it can be shown that
$$\hat{f}(\boldsymbol{x}) = \bar{y} + \widetilde{\boldsymbol{y}}^T\left(\tilde{G} + n\lambda I\right)^{-1}\widetilde{\boldsymbol{g}}(\boldsymbol{x})$$
where
$$\tilde{G} = \begin{bmatrix} \langle \widetilde{\boldsymbol{x}}_1, \widetilde{\boldsymbol{x}}_1 \rangle & \dots & \langle \widetilde{\boldsymbol{x}}_1, \widetilde{\boldsymbol{x}}_n \rangle \\ \vdots & & \vdots \\ \langle \widetilde{\boldsymbol{x}}_n, \widetilde{\boldsymbol{x}}_1 \rangle & \dots & \langle \widetilde{\boldsymbol{x}}_n, \widetilde{\boldsymbol{x}}_n \rangle \end{bmatrix}, \qquad \widetilde{\boldsymbol{g}}(\boldsymbol{x}) = \begin{bmatrix} \langle \widetilde{\boldsymbol{x}}_1, \boldsymbol{x} - \overline{\boldsymbol{x}} \rangle \\ \vdots \\ \langle \widetilde{\boldsymbol{x}}_n, \boldsymbol{x} - \overline{\boldsymbol{x}} \rangle \end{bmatrix}$$
- Now suppose we first transform the data with $\boldsymbol{\Phi}$
- It's tempting to substitute $\langle \widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{x}}' \rangle$ with $\langle \boldsymbol{\Phi}(\widetilde{\boldsymbol{x}}), \boldsymbol{\Phi}(\widetilde{\boldsymbol{x}}') \rangle$

KRR with Offset

- To kernelize, observe

$$\langle \tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j \rangle = \langle \boldsymbol{x}_i - \overline{\boldsymbol{x}}, \boldsymbol{x}_j - \overline{\boldsymbol{x}} \rangle$$
$$= \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \frac{1}{n}\sum_{r=1}^{n} \langle \boldsymbol{x}_i, \boldsymbol{x}_r \rangle - \frac{1}{n}\sum_{s=1}^{n} \langle \boldsymbol{x}_s, \boldsymbol{x}_j \rangle + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} \langle \boldsymbol{x}_r, \boldsymbol{x}_s \rangle$$

And

$$\langle \tilde{\boldsymbol{x}}_i, \boldsymbol{x} - \overline{\boldsymbol{x}} \rangle = \langle \boldsymbol{x}_i - \overline{\boldsymbol{x}}, \boldsymbol{x} - \overline{\boldsymbol{x}} \rangle$$
$$= \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle - \frac{1}{n}\sum_{r=1}^{n} \langle \boldsymbol{x}_i, \boldsymbol{x}_r \rangle - \frac{1}{n}\sum_{s=1}^{n} \langle \boldsymbol{x}, \boldsymbol{x}_s \rangle + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} \langle \boldsymbol{x}_r, \boldsymbol{x}_s \rangle$$

- Now replace $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ with $k(\boldsymbol{x}, \boldsymbol{x}')$ throughout the above expressions

KRR with Offset: alternative View

- Define the "mean-centered feature map"

$$\tilde{\boldsymbol{\Phi}}(\boldsymbol{x}) := \boldsymbol{\Phi}(\boldsymbol{x}) - \frac{1}{n}\sum_{\ell=1}^{n} \boldsymbol{\Phi}(\boldsymbol{x}_\ell)$$

- KRR with offset can equivalently be expressed

$$\widehat{f}(\boldsymbol{x}) = \bar{y} + \tilde{\boldsymbol{y}}^T(\tilde{\boldsymbol{K}} + n\lambda \boldsymbol{I})^{-1}\tilde{\boldsymbol{k}}(\boldsymbol{x})$$

where the (i, j) entry of $\tilde{\boldsymbol{K}}$ is

$$\langle \tilde{\Phi}(x_i), \tilde{\Phi}(x_j) \rangle$$

and the i-th entry of $\tilde{\boldsymbol{k}}(\boldsymbol{x})$ is

$$\langle \tilde{\Phi}(x_i), \tilde{\Phi}(x) \rangle$$

- These values are computed using the formulas on the previous slide, after replacing dot products with kernels.