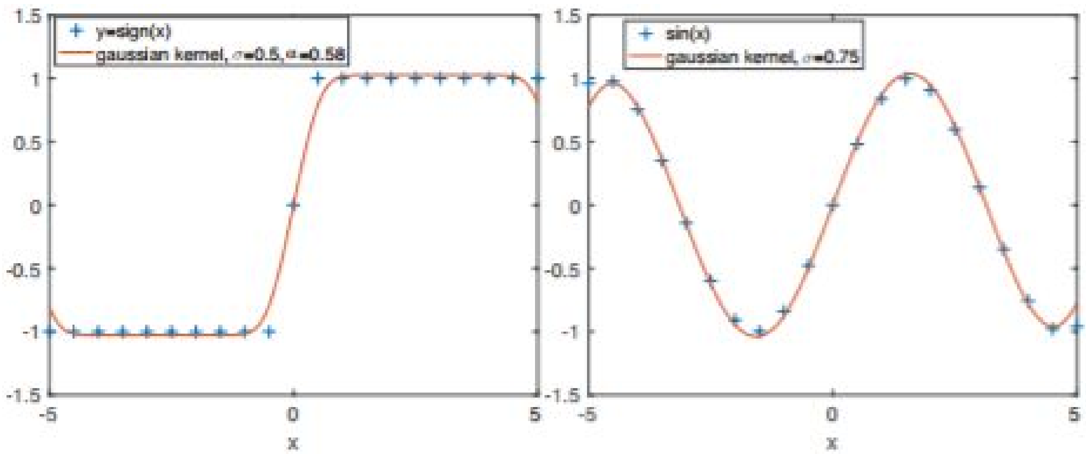
Kernel Methods

- **Instance-based Learning: kNN**
    - **Training data:** $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})\}$
    - Given a new observation x, find the nearest point in the training data e.g. $x^{(2)}$ and use its label $y^{(2)}$ as the prediction
    - **Generalise to finding k nearest neighbours and predicting the label by majority vote**
        - **This idea applies to both real valued outputs and classification**
    - How to measure distance?
        - **X is a vector**
        - **Often use Euclidean distance** $\sum_{j=1}^{n} x_j^2$

- **Instance-based Learning Using Kernels**
    - Define a weighting function K(x,z) that is a maximum when x=z and decays as the distance between x and z increases
    - **This is a kernel.** Requires K(x,z) to be of the form $\Phi(x)^T \phi(z) = \Phi(xz\phi(x)$ for some mapping ϕ
    - **E.g. Gaussian kernel**
        - **K(x,z) =** $e^{\frac{\sum_{j=1}^{n}(x_j - z_j)^2}{\sigma^2}}$
        - Parameter σ controls how quickly the weighting decays (i.e. the width of the bell shape)
    - Use prediction: $sign(\sum_{i=1}^{m} \alpha_i y^{(i)} K(x, x^{(i)}))\ where\ \alpha_i \geq 0,\ i = 1, ..., m\ are\ parameters\ to\ be\ chosen$



    -
    - Plot is of $\sum_{i=1}^{m} \alpha y^{(i)} K(x, x^{(i)})\ with\ K(x, x^{(i)}) = e^{\frac{(x - x^{(i)})^2}{\sigma^2}}$

- Notice the edge effects (when no training data, the prediction reverts to zero)
- **Increasing σ makes the kernel broader (underfits data), decreasing σ makes the kernel narrower (overfitting)**

- **Kernel Logistic Regression**
    - **Replace $\theta^T x$ with $\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)})$**
    - **Hypothesis:** $sign(\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)}))$
    - **Cost:** $J(\alpha) = \frac{1}{m} \sum_{i=1}^{m} log(1 + e^{-y^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})})$
    - Use gradient descent to select α , select σ and λ using cross-validation
    - **Kernels provide an alternative way to handle nonlinear decision boundaries**
        - More flexible, but more expensive and prone to overfitting

- **Kernel SVMs**
    - Replace $\theta^T x$ with $\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)})$
    - **Hypothesis:** $sign(\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)}))$
    - **Cost:** $J(\alpha, \theta) = \frac{1}{m} \sum_{i=1}^{m} max(0, 1 - y^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})) + \lambda\theta^T\theta$
    - **What about $\theta^T\theta$? We would like cost in terms of α**
        - What we've done so far is replace x with Φ(x)
        - Changing $\theta^T x \text{ to } \theta^T \Phi(x)$ define $\theta = \sum_{j=1}^{m} \alpha_j y^{(j)} \Phi(x^{(j)})$
        - Then $\theta^T \Phi(x) = \sum_{j=1}^{m} \alpha_j y^{(j)} \Phi(x^{(j)})^T \Phi(x)$ = $\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)})$
            - $\theta^T \theta = \sum_{j=1}^{m} \alpha_j y^{(j)} \Phi(x^{(j)})^T \sum_{i=1}^{m} \alpha_i y^{(i)} \Phi(x^{(i)})$
            - = $\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(j)}, x^{(i)}) y^{(i)} \alpha_i = \alpha^T M \alpha$
            - **Where M is matrix with $M_{ij} = y^{(j)} K(x^{(j)}, x^{(i)}) y^{(i)}$ and α is parameter vector**
        - **Cost:**
            - $J(\alpha) = \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})) + \lambda \alpha^T M \alpha$
    - **Summary of Kernel SVMs:**

Hypothesis: $sign(\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)}))$

Cost:

$J(\alpha) = \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})) + \lambda \alpha^T M \alpha$

Use gradient descent to select α as usual. Select σ (kernel parameter) and λ using cross-validation.

- **Kernel Ridge Regression**

Replace $\theta^T x$ with $\sum_{j=1}^{m} \alpha_j y^{(i)} K(x, x^{(j)})$

Use $\theta^T \theta = \alpha^T M \alpha$ where M is matrix with $M_{ij} = y^{(j)} K(x^{(j)}, x^{(i)}) y^{(i)}$ and α is parameter vector.

Hypothesis: $\sum_{j=1}^{m} \alpha_j y^{(i)} K(x, x^{(j)})$

Cost: $J(\alpha) = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - \sum_{j=1}^{m} \alpha_j y^{(i)} K(x^{(i)}, x^{(j)}))^2 + \lambda \alpha^T M \alpha$

Use gradient descent to select α as usual, or can use closed-form solution. Select σ (kernel parameter) and λ using cross-validation.

Use of kernels provides another way to fit nonlinear curves.

Regression with a Gaussian kernel is also known as Radial Basis Function Regression (or sometimes as a Radial Basis Function Network)