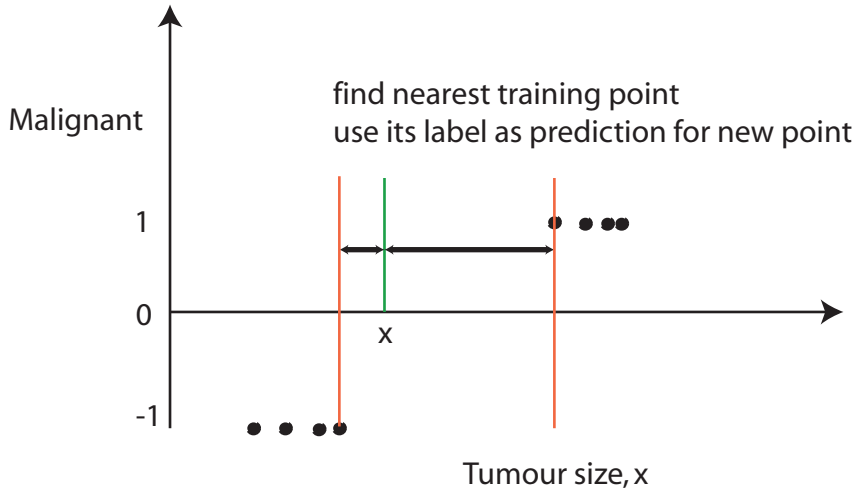
Overview

- Instance-based Learning: kNN
- Instance-based Learning Using Kernels
- Kernel Logistic Regression and Kernel SVMs
- Kernel Regression

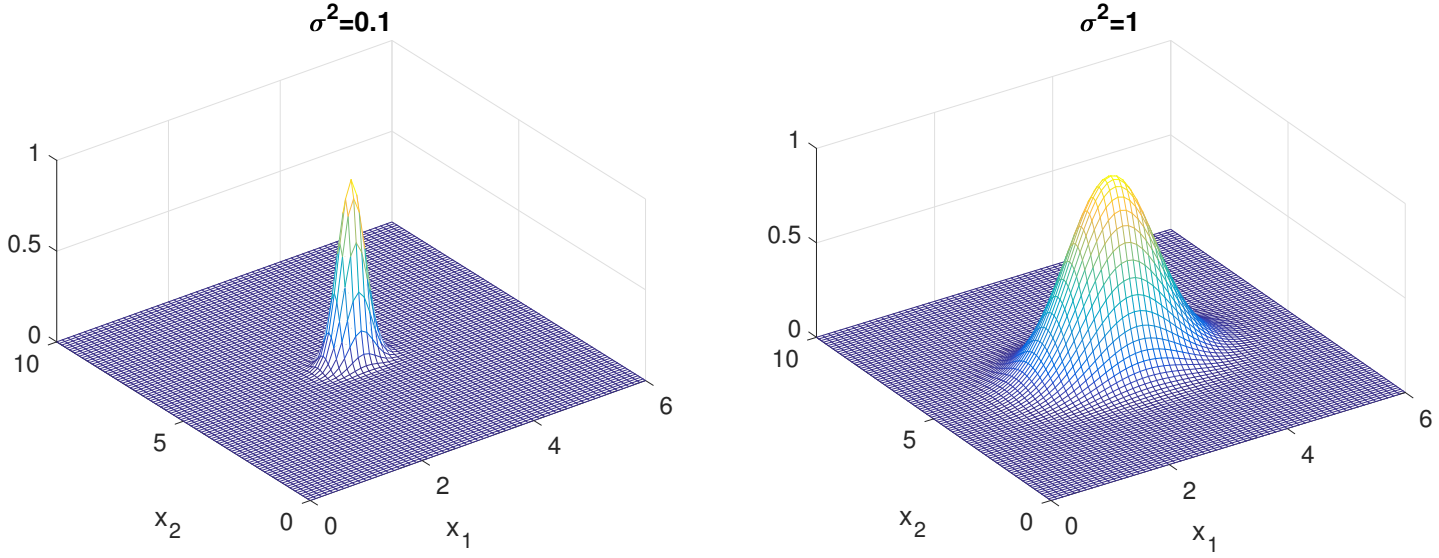
Instance-based Learning: kNN



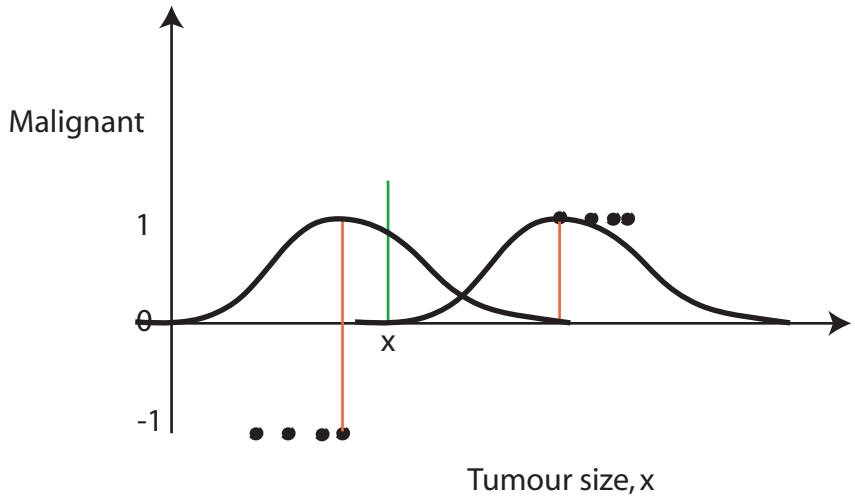
- Training data: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$
- Given new input x, find the nearest point in the training data e.g. $x^{(2)}$ and use its label e.g. $y^{(2)}$ as our prediction
- Can generalise to finding the k nearest neighbours (hence kNN) and predicting the label by majority vote
- We can use the same idea for both real valued outputs and for classification.
- How do we measure distance i.e. decide which points are nearest ? Recall 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 maximum when $x = z$ and decays as the distance between x and z increases.
- This function is often called a **kernel**, although this requires $K(x, z)$ to be of the form $\phi(x)^T\phi(z) = \phi(z)^T\phi(x)$ 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_i controls how quickly the weighting decays i.e how narrow or wide the bell shape is.
- Example: $z = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$

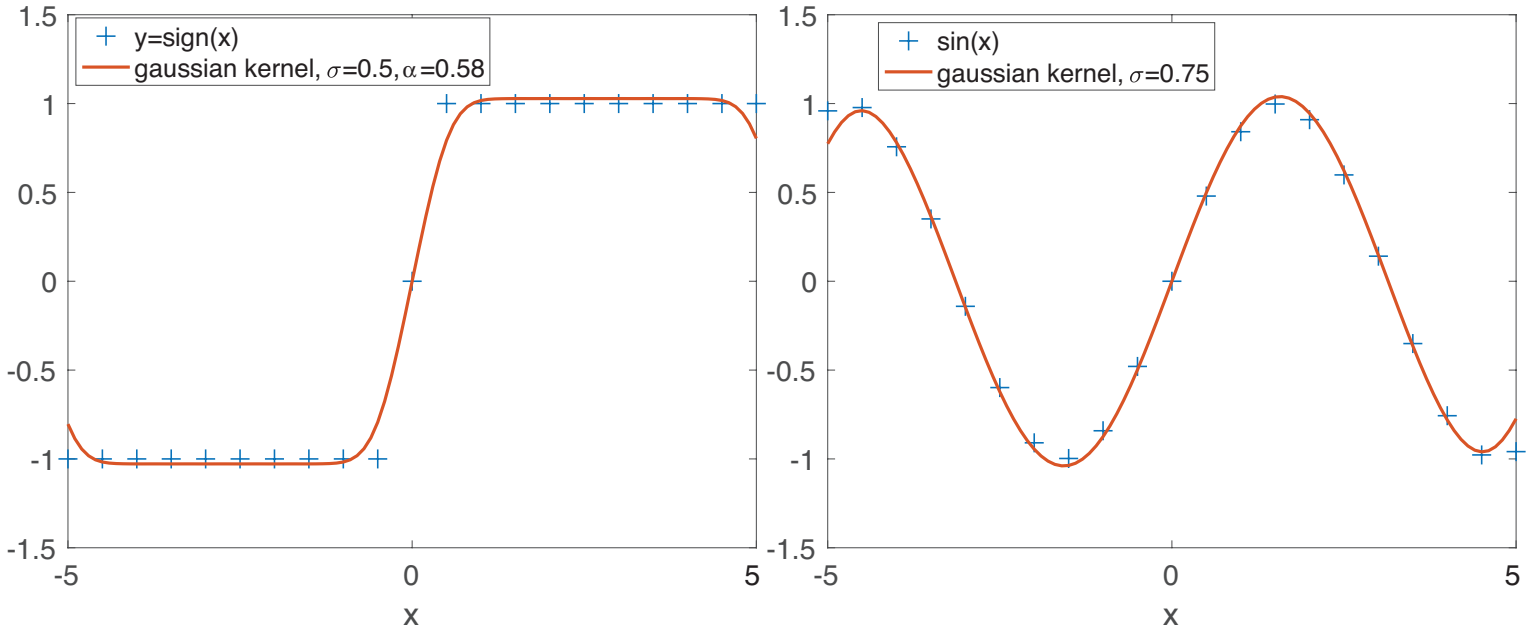


Instance-based Learning Using Kernels



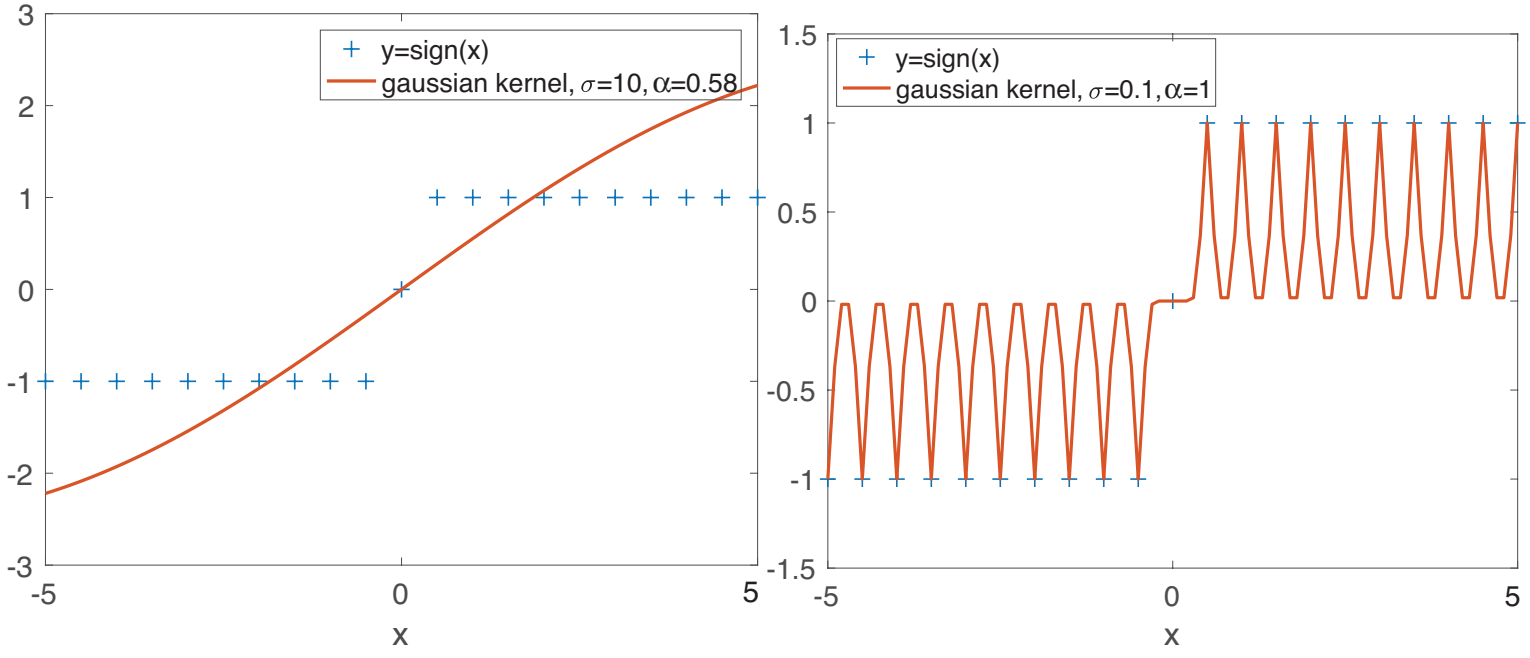
- Use prediction $sign(\sum_{i=1}^{m} \alpha_i y^{(i)} K(x, x^{(i)}))$ where $\alpha_i \geq 0$, $i = 1, \ldots, m$ are parameters to be chosen.

Instance-based Learning Using Kernels



- Plot is of $\sum_{i=1}^{m} \alpha y^{(i)} K(x, x^{(i)})$ with $K(x, x^{(i)}) = e^{\frac{(x-x^{(i)})^2}{\sigma^2}}$ (use same α and σ values for all points i).
- Notice the edge effects. When no training data then with Gaussian kernel prediction reverts to zero.

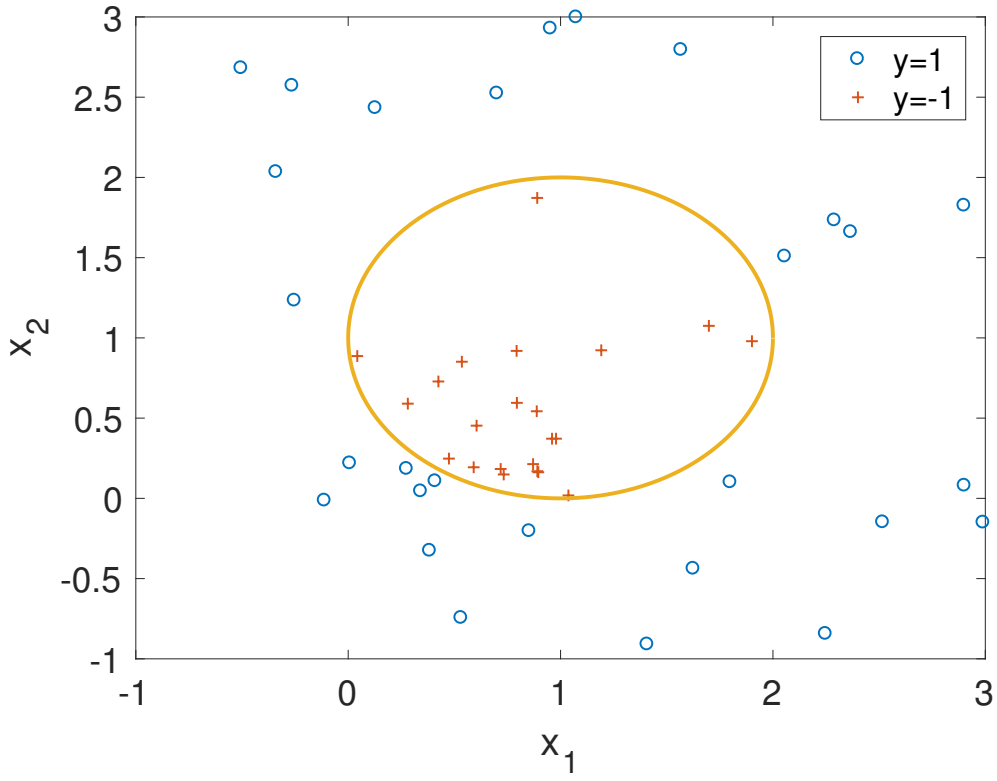
Instance-based Learning Using Kernels



- Increasing σ makes the kernel broader. Tends to underfit the training data points.
- Decreasing σ makes the kerner narrower. Tends to overfit the training data points.

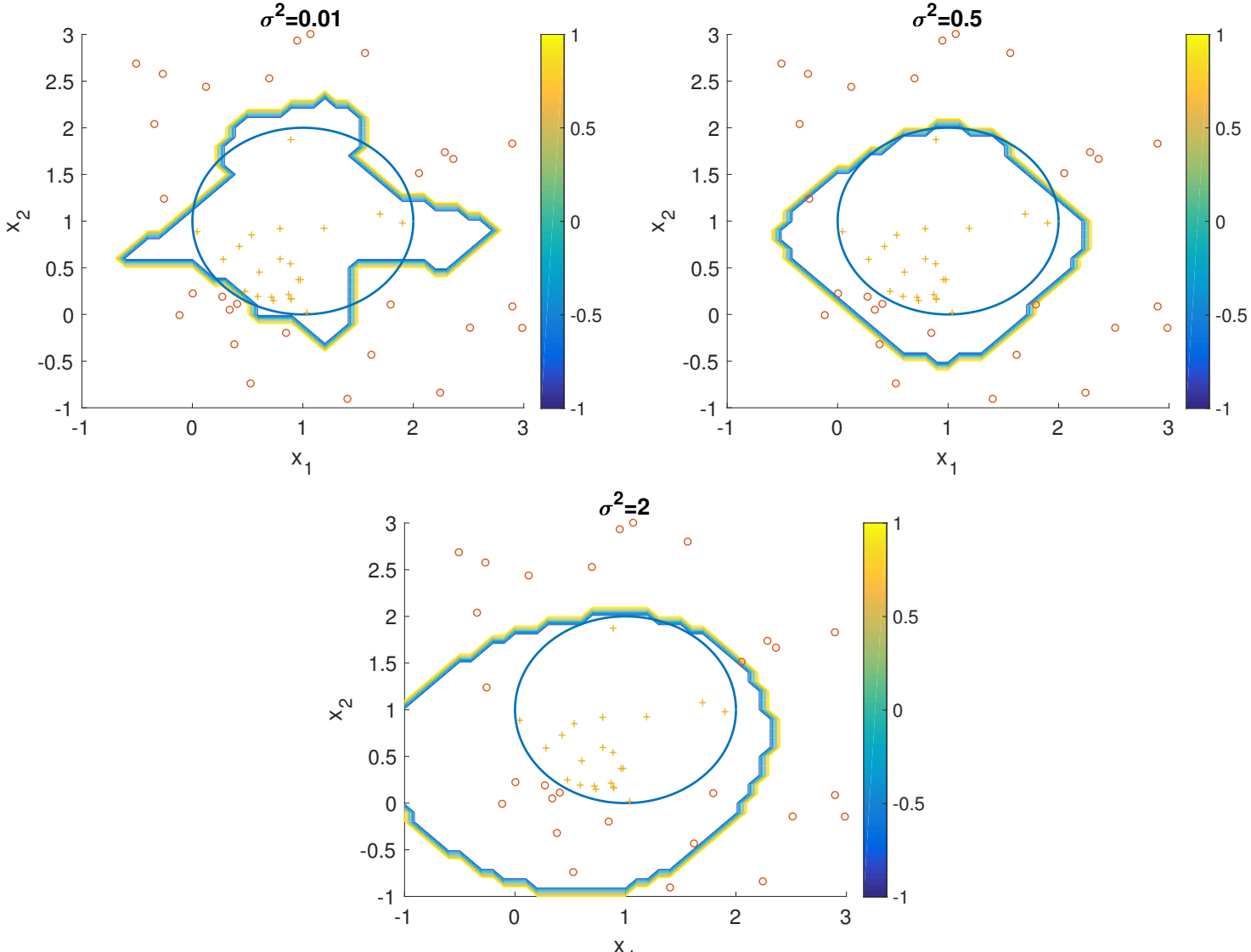
Example: Nonlinear Decision Boundary

Generate training data using $y = sign((x_1 - 1)^2 + (x_2 - 1)^2 - 1)$.



Example: Nonlinear Decision Boundary

Fit $sign(\sum_{j=1}^{m} \alpha y^{(j)} K(x, x^{(j)}))$, using Gaussian kernel with $\alpha = 0.5$ and various values of σ^2.



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: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \log(1 + e^{-y^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})})$
- Use gradient descent to select θ as usual. Select σ (kernel parameter) and λ using cross-validation.
- Use of kernels provides another way to handle nonlinear decision boundaries. The extra flexibility comes at greater computational cost (more parameters to choose) and with risk of overfitting.

Kernel SVMs[1]

- Replace $\theta^T x$ with $\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)})$ as before
- Hypothesis: $sign(\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)}))$
- Cost:
  $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})) + \lambda \theta^T \theta$
- What about $\theta^T \theta$ term ? We'd like cost to be only in terms of α

[1] Training a Support Vector Machine in the Primal. Olivier Chapelle, Neural Computation 2007

Kernel SVMs

What about $\theta^T\theta$ term ?

- There's another way to think about kernel approaches. What we're doing is replacing x by $\phi(x)$, generalising what we did when we used polynomials to fit nonlinear decision boundaries.
- Changing $\theta^T x$ 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:
  $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})) + \lambda \alpha^T M \alpha$

Kernel SVMs: Summary

- Hypothesis: $sign(\sum_{j=1}^{m} \alpha_j y^{(j)} K(x, x^{(j)}))$
- Cost:
  $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})) + \lambda \alpha^T M \alpha$
- Use gradient descent to select α as usual. Select σ (kernel parameter) and λ using cross-validation.

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