Machine learning - HT 2016
7. Classification: Support Vector Machines

Varun Kanade

University of Oxford
February 12, 2016

Announcements

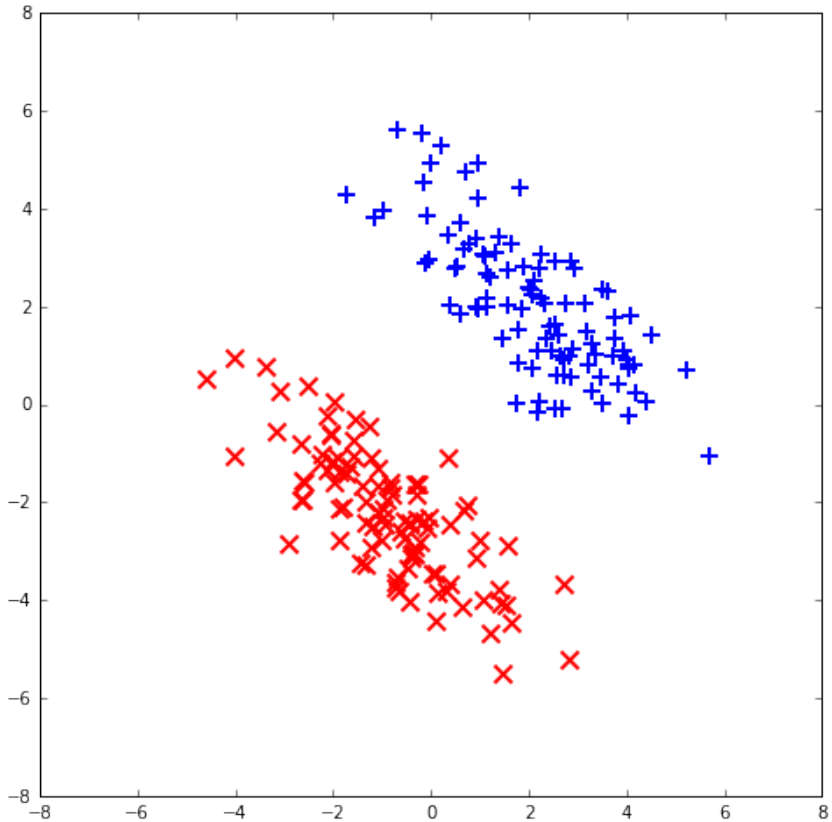
- Problem Sheet 4 due next Friday by noon
- Reading a paper

Outline

Today we'll discuss classification using support vector machines.

- No clear probabilistic interpretation
- Maximum Margin Formulation
- Optimisation problem using Hinge Loss
- Dual Formulation

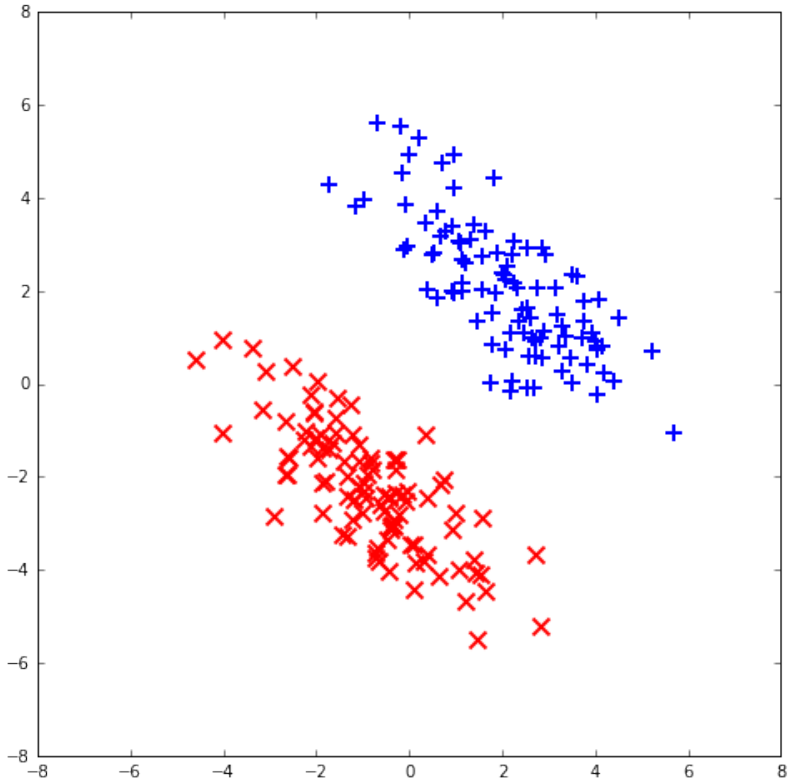
Binary Classification

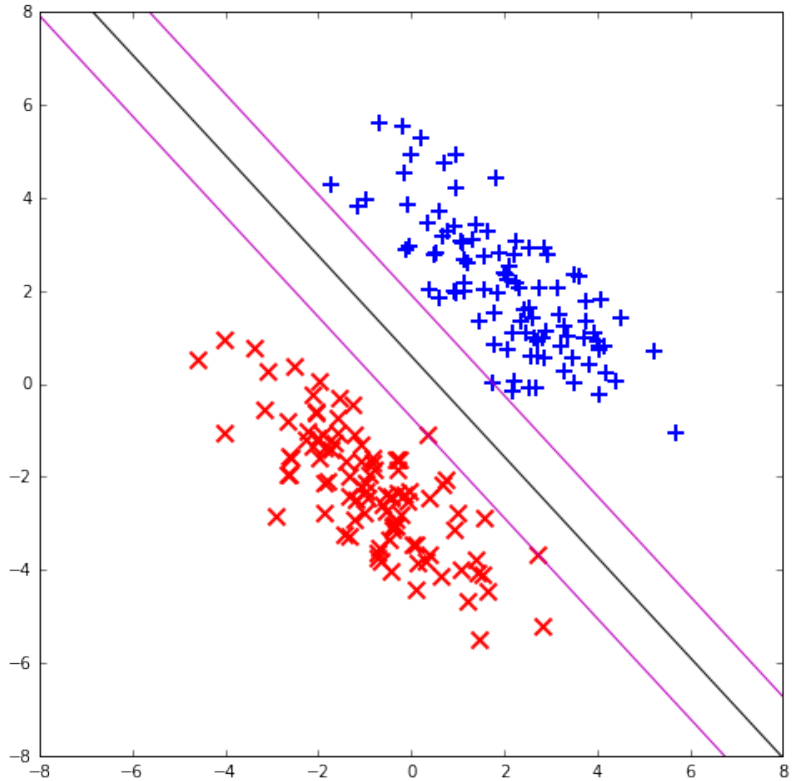


Goal: Find a linear separator

Data is linearly separable if there exists a linear separator with no classification error

Maximum Margin Principle





Maximise the distance of the closest point from the decision boundary

Geometry

Given a hyperplane: $H \equiv \mathbf{w} \cdot \mathbf{x} + w_0 = 0$ and a point $\mathbf{x} \in \mathbb{R}^n$

How far is $\mathbf{x}$ from H?

Geometry

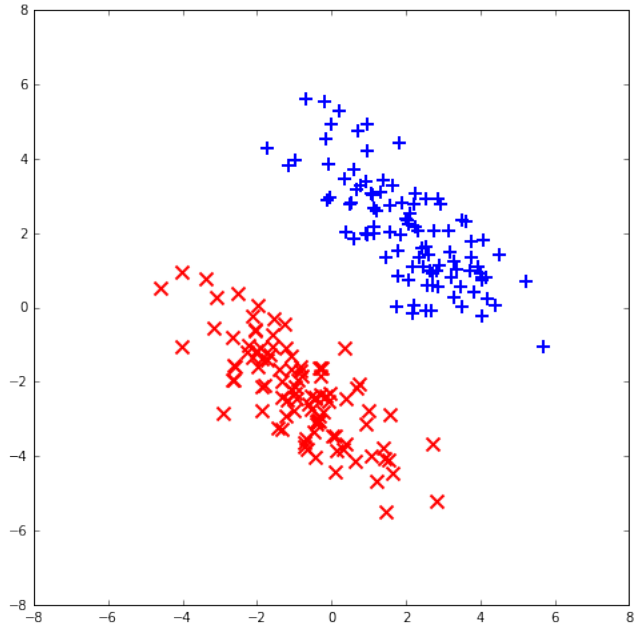
For a hyperplane: $H = \mathbf{w} \cdot \mathbf{x} + w_0 = 0$ with $\|\mathbf{w}\|_2 = 1$

The distance of point $\mathbf{x}$ from H is $|\mathbf{w} \cdot \mathbf{x} + w_0|$

If we don't restrict $\|\mathbf{w}\|_2$ to be 1, then the distance is:

$$\frac{|\mathbf{w} \cdot \mathbf{x} + w_0|}{\|\mathbf{w}\|_2}$$

SVM Formulation : Separable Case



SVM Formulation : Separable Case

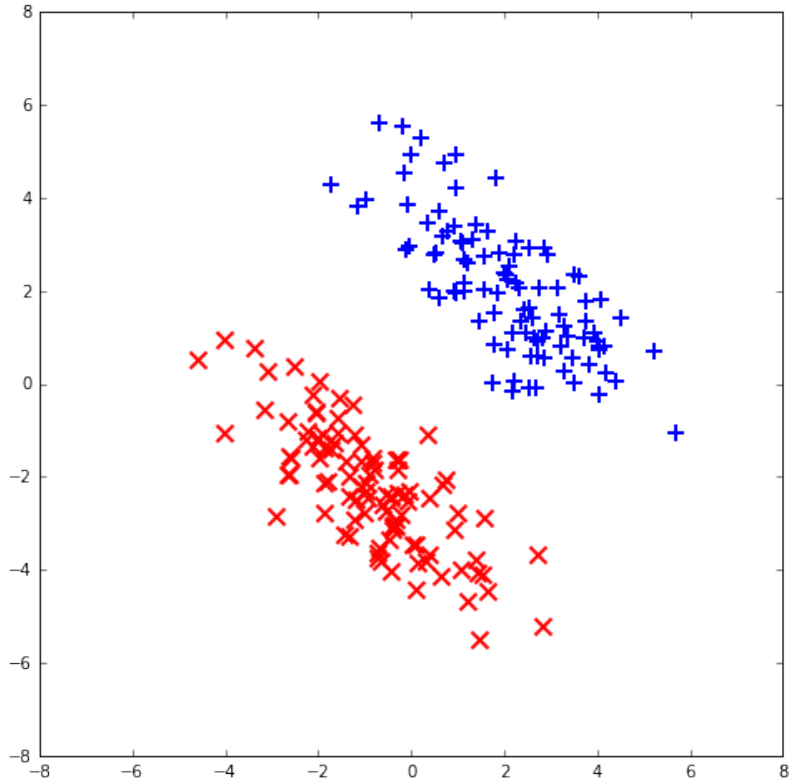
minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2$

subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1$$

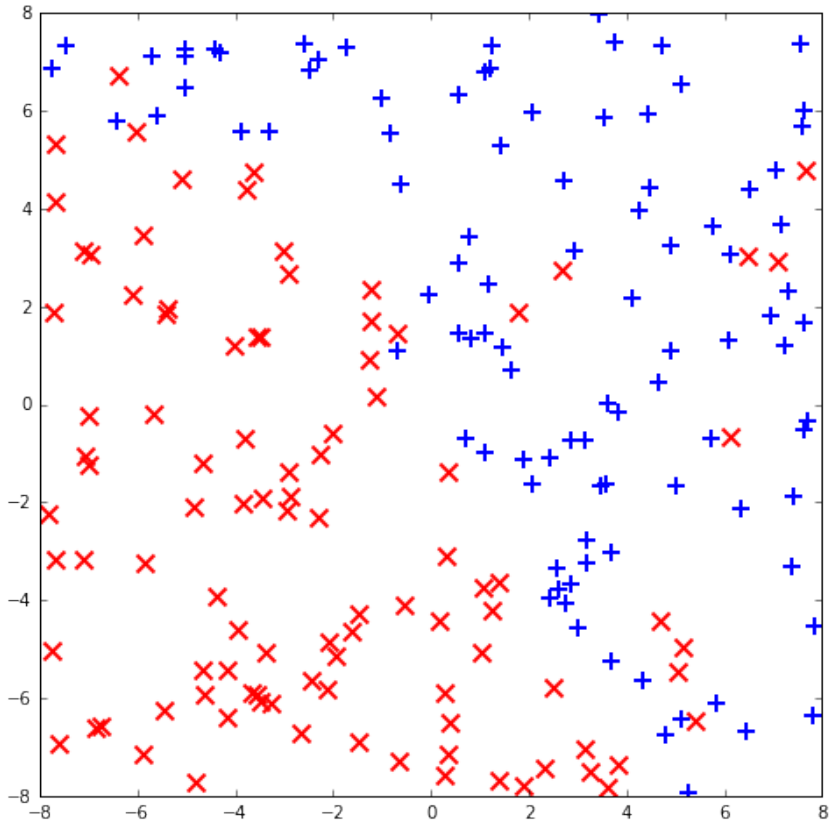
for $i = 1, \ldots, m$

Here $y_i \in \{-1, 1\}$



If data is separable, then we find a classifier with no classification error on the training set

Non-separable Data



Quadratic program on previous slide has no feasible solution

Which linear separator should we try to find?

SVM Formulation : Non-Separable Case

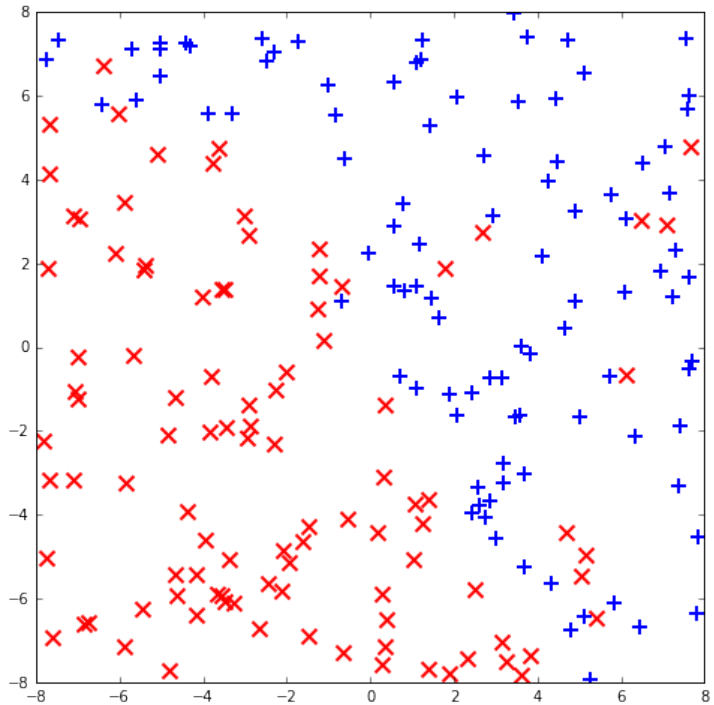
minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2$

subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1$$

for $i = 1, \ldots, m$

Here $y_i \in \{-1, 1\}$



SVM Formulation : Non-Separable Case

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\zeta_i$

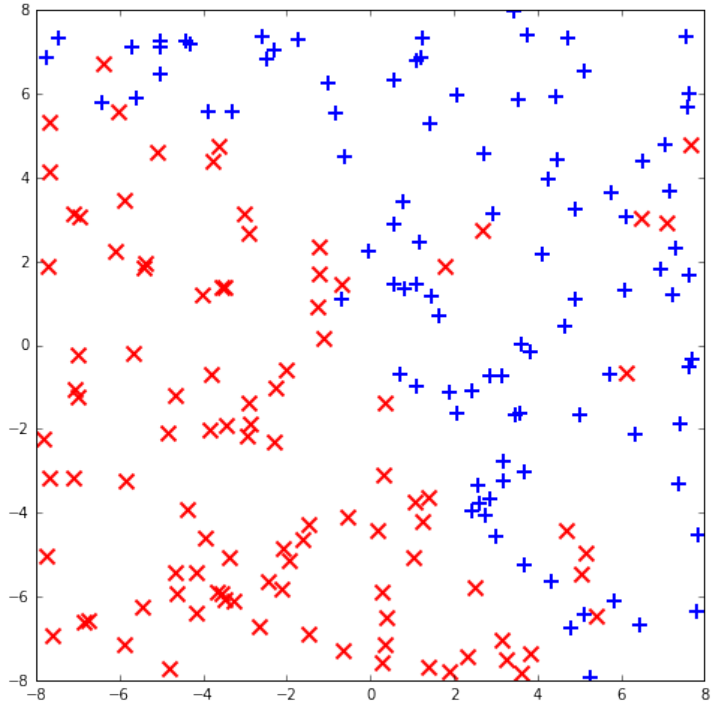
subject to:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 - \zeta_i$$

$$\zeta_i \geq 0$$

for $i = 1, \ldots, m$

Here $y_i \in \{-1, 1\}$



SVM Formulation : Loss Function

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\zeta_i$

subject to:

$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 - \zeta_i$

$\zeta_i \geq 0$

for $i = 1, \ldots, m$

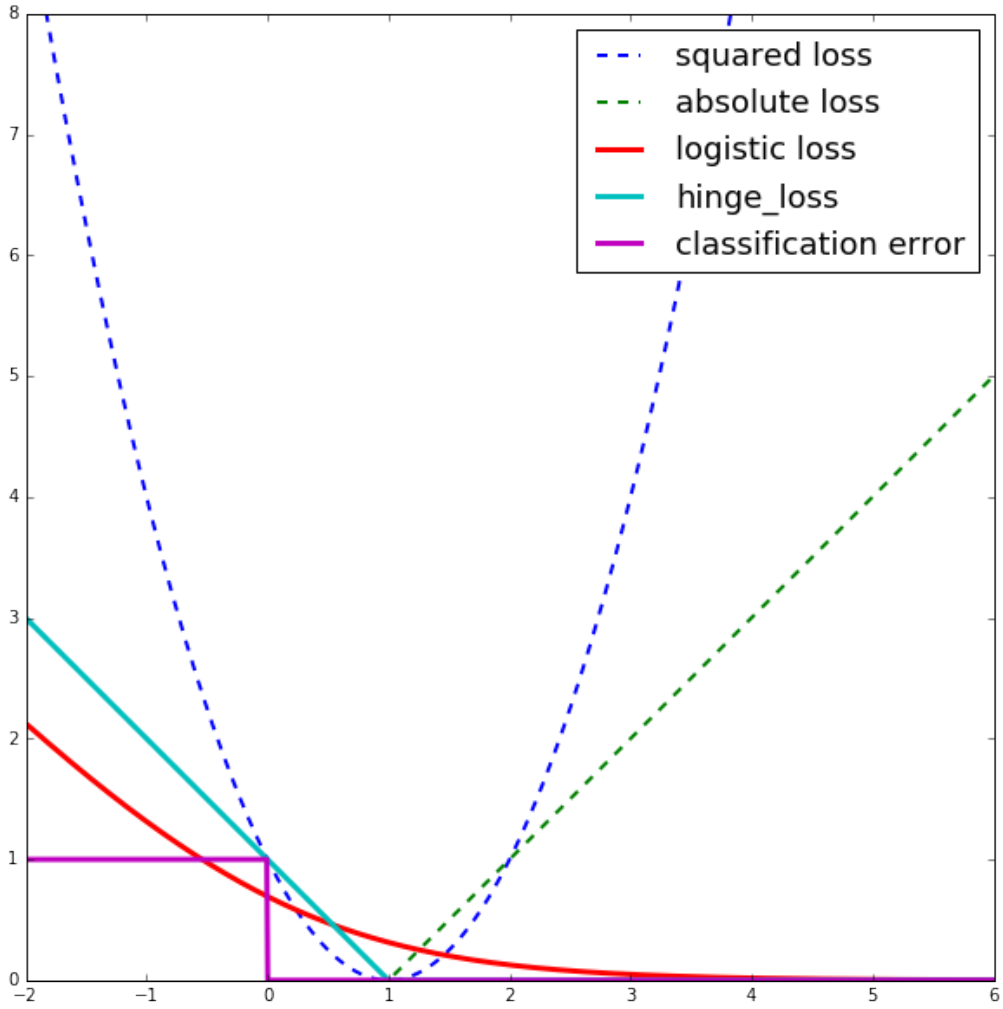
Here $y_i \in \{-1, 1\}$

Logistic Regression: Loss Function

Here $y_i \in \{0, 1\}$

$$\ell(\mathbf{w}; \mathbf{x}_i, y_i) = -\left(y_i \log\left(\frac{1}{1 + e^{-\mathbf{w}\cdot\mathbf{x}_i}} \right) + (1 - y_i) \log\left(\frac{1}{1 + e^{\mathbf{w}\cdot\mathbf{x}_i}} \right) \right)$$

Loss Functions



SVM Formulation : Non-Separable Case

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\zeta_i$

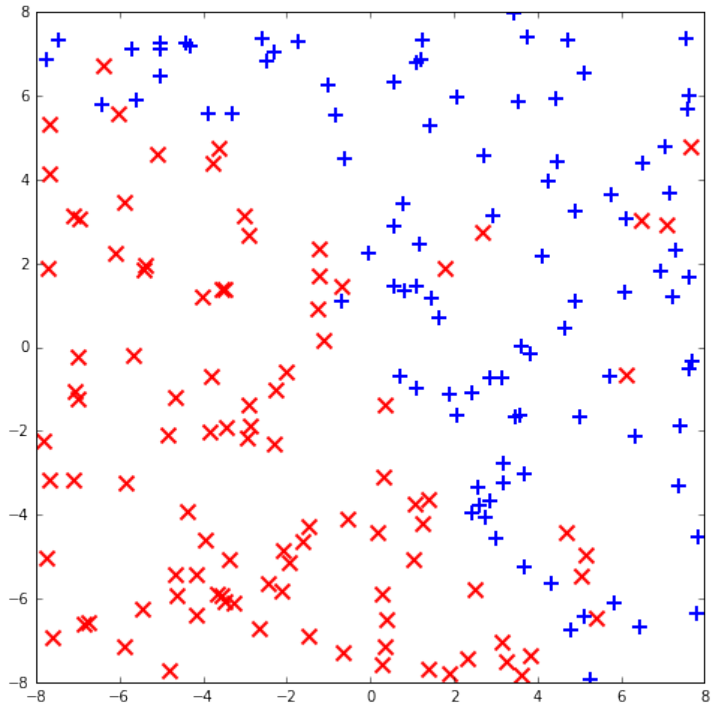
subject to:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 - \zeta_i$$

$$\zeta_i \geq 0$$

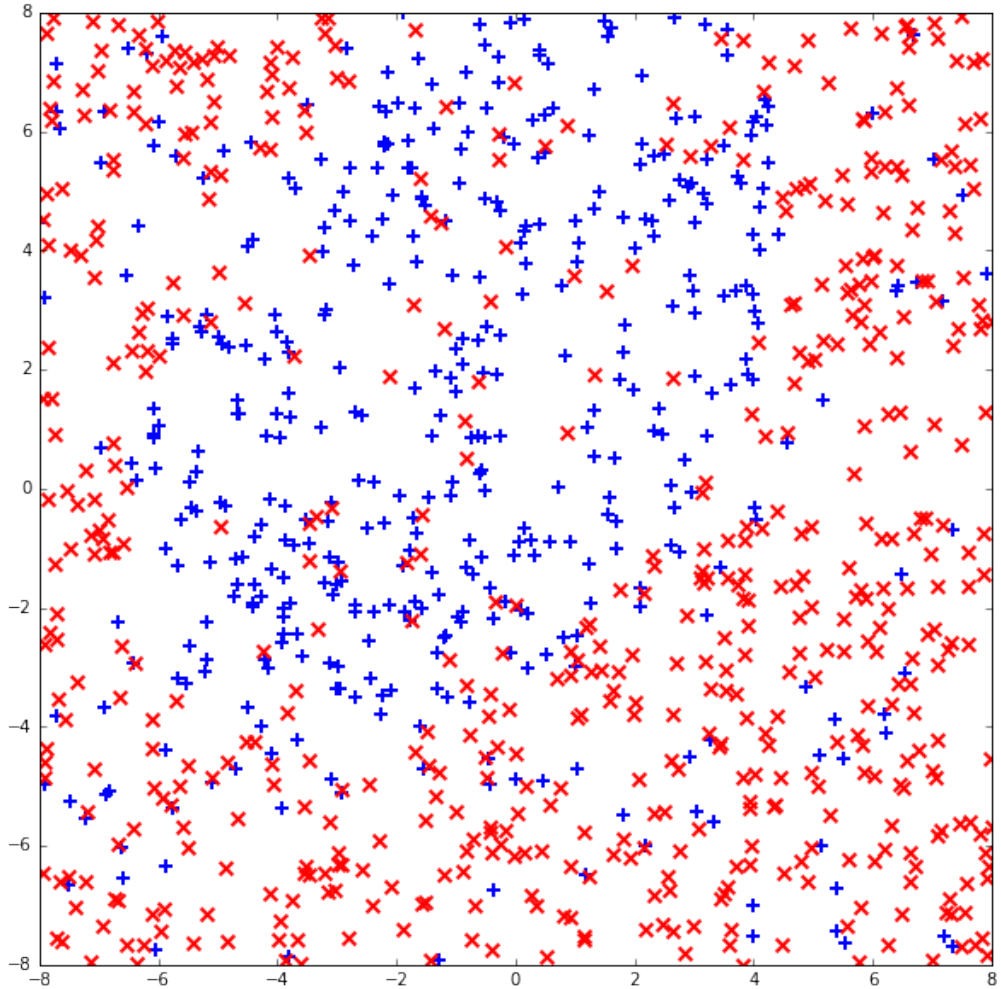
for $i = 1, \ldots, m$

Here $y_i \in \{-1, 1\}$



SVM Formulation: Non-Separable Case

What if your data looks like this?



SVM Formulation : Constrained Minimisation

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\zeta_i$

subject to:

$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i) \geq 0$

$\zeta_i \geq 0$

for $i = 1, \ldots, m$

Here $y_i \in \{-1, 1\}$

SVM Formulation

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\zeta_i$

subject to:

$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i) \geq 0$

$\zeta_i \geq 0$

for $i = 1, \ldots, m$

Here $y_i \in \{-1, 1\}$

Lagrange Function

$$L(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i - \sum_{i=1}^{m}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i)) - \sum_{i=1}^{m}\mu_i\zeta_i$$

Dual Formulation

(Idealised) Lagrange Function

$$L_{\text{ideal}}(\mathbf{w}, w_0, \boldsymbol{\zeta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i - \sum_{i=1}^{m}\mathbb{I}_{\geq 0}(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i)) - \sum_{i=1}^{m}\mathbb{I}_{\geq 0}(\zeta_i),$$

where $\mathbb{I}_{\geq 0}(x) = 0$ if $x \geq 0$ and $-\infty$ otherwise.

Dual Formulation

(Idealised) Lagrange Function

$$L_{\text{ideal}}(\mathbf{w}, w_0, \boldsymbol{\zeta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i - \sum_{i=1}^{m}\mathbb{I}_{\geq 0}(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i)) - \sum_{i=1}^{m}\mathbb{I}_{\geq 0}(\zeta_i),$$

where $\mathbb{I}_{\geq 0}(x) = 0$ if $x \geq 0$ and $-\infty$ otherwise.

Lagrange Function

$$L(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i - \sum_{i=1}^{m}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i)) - \sum_{i=1}^{m}\mu_i\zeta_i$$

For any $\boldsymbol{\alpha}, \boldsymbol{\mu} \geq 0$, we have for all $\mathbf{w}, w_0$

$$L(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) \leq L_{\text{ideal}}(\mathbf{w}, w_0, \boldsymbol{\zeta})$$

Dual Formulation

Lagrange Function

$$L(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i - \sum_{i=1}^{m}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i)) - \sum_{i=1}^{m}\mu_i\zeta_i$$

$$g(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \inf_{\mathbf{w}, w_0} L(\mathbf{w}, w_0; \boldsymbol{\alpha}, \boldsymbol{\mu})$$

Dual Formulation

Lagrange Function

$$L(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i - \sum_{i=1}^{m}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i)) - \sum_{i=1}^{m}\mu_i\zeta_i$$

$$g(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \inf_{\mathbf{w}, w_0} L(\mathbf{w}, w_0; \boldsymbol{\alpha}, \boldsymbol{\mu})$$

Maximising $g(\boldsymbol{\alpha}, \boldsymbol{\mu})$ will give us the "best possible" lower bound

Dual Formulation

Lagrange Function

$$L(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i - \sum_{i=1}^{m}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i)) - \sum_{i=1}^{m}\mu_i\zeta_i$$

Dual Formulation

Primal Form

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2$

subject to:

$y_i(\mathbf{w}\cdot\mathbf{x}_i+w_0) \geq 1-\zeta_i$

$\zeta_i \geq 0$

for $i = 1, \ldots, m$

Dual Form

maximise: $\sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$

subject to:

$C \geq \alpha_i \geq 0$

$\sum_{i=1}^{m} \alpha_i y_i = 0$

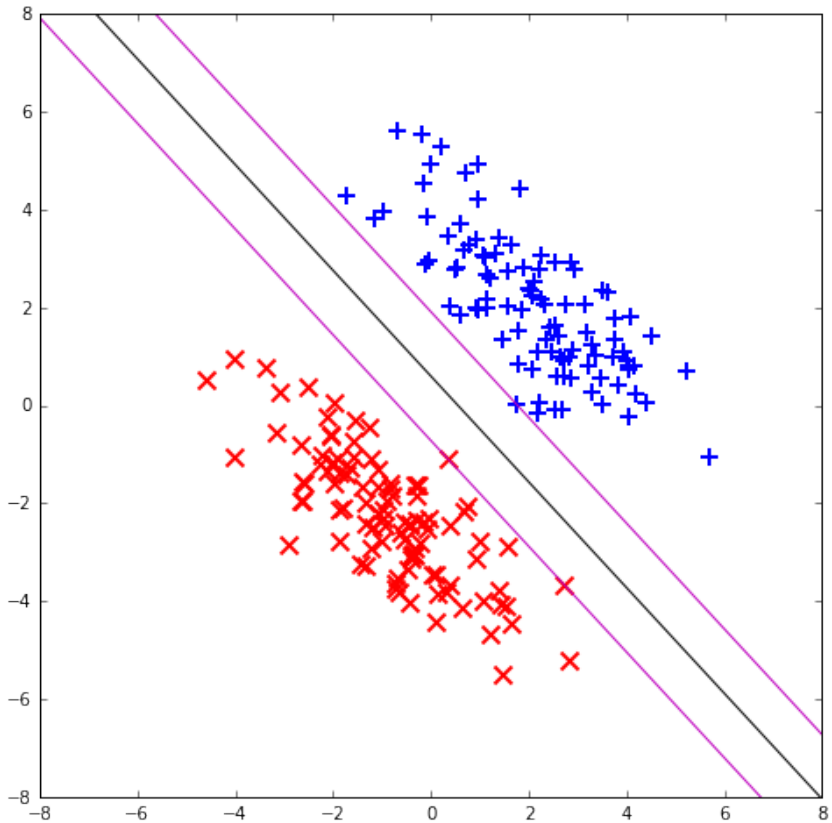
Lagrange Function

$$L(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\zeta_i - \sum_{i=1}^{m}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i+w_0)-(1-\zeta_i)) - \sum_{i=1}^{m}\mu_i\zeta_i$$

KKT Conditions

$\alpha_i^* = 0$ or $y_i(\mathbf{w}^* \cdot \mathbf{x}_i + w_0^*) - 1 = 0$, where $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i$

Support Vectors



SVM Dual Program

maximise: $$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to:

$$C \geq \alpha_i \geq 0$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

SVM Dual Program

maximise: $$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to:

$$C \geq \alpha_i \geq 0$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

- Objective depends only between dot products of training inputs
- Particularly useful if inputs are high-dimensional
- To make a new prediction only need to know dot product with support vectors

Gram Matrix

If we put the inputs in matrix $\mathbf{X}$, where the i^{th} row of $\mathbf{X}$ is $\mathbf{x}_i^T$.

$$\mathbf{K} = \mathbf{X}\mathbf{X}^T = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_m \rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1 \rangle & \langle \mathbf{x}_2, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_m \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{x}_m, \mathbf{x}_1 \rangle & \langle \mathbf{x}_m, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_m, \mathbf{x}_m \rangle \end{bmatrix}$$

- The matrix K is positive definite (if $n > m$ and $\mathbf{x}_i$ are linearly independent)
- If we do feature expansion first

$$\phi : \mathbb{R}^n \to \mathbb{R}^N$$

then replace entries by $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$

Kernel Trick

Suppose we do basis expansion with up to degree two terms (say $n = 2$)

$$\phi((x_1, x_2)) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Kernel Trick

Suppose we do basis expansion with up to degree two terms (say $n = 2$)

$$\phi((x_1, x_2)) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

If $\mathbf{x} = (x_1, x_2)$ and $\mathbf{x}' = (x_1', x_2')$ then what is $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$?

Kernel Trick

We can use a symmetric positive definite matrix (Mercer Kernels)

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_m) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_m, \mathbf{x}_1) & \kappa(\mathbf{x}_m, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}$$

Here $\kappa(\mathbf{x}, \mathbf{z})$ is some measure of similarity between $\mathbf{x}$ and $\mathbf{x}'$

Kernel Trick

We can use a symmetric positive definite matrix (Mercer Kernels)

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_m) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_m, \mathbf{x}_1) & \kappa(\mathbf{x}_m, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}$$

Here $\kappa(\mathbf{x}, \mathbf{z})$ is some measure of similarity between $\mathbf{x}$ and $\mathbf{x}'$

The dual program becomes

$$\text{minimise} \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K_{i,j} \qquad \text{subject to} : 0 \leq \alpha_i \leq C$$

Kernel Trick

We can use a symmetric positive definite matrix (Mercer Kernels)

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_m) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_m, \mathbf{x}_1) & \kappa(\mathbf{x}_m, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}$$

Here $\kappa(\mathbf{x}, \mathbf{z})$ is some measure of similarity between $\mathbf{x}$ and $\mathbf{x}'$

The dual program becomes

$$\text{minimise} \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K_{i,j} \qquad \text{subject to} : 0 \leq \alpha_i \leq C$$

To make prediction on new $\mathbf{x}_{\text{new}}$, need to compute $\kappa(\mathbf{x}_i, \mathbf{x}_{\text{new}})$ for support vectors $\mathbf{x}_i$

Polynomial Kernels

Rather than perform basis expansion,

$$\kappa(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x} \cdot \mathbf{x}')^d$$

This gives all terms of degree up to d

If we use $\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^d$, we get only degree d terms

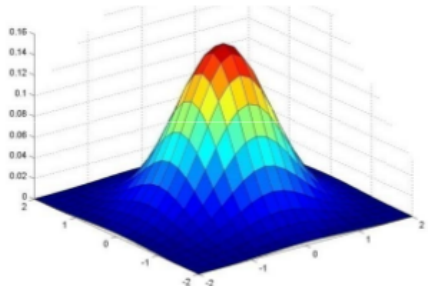
Linear Kernel: $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$

All of these satisfy the Mercer or positive-definite condition

Gaussian or RBF Kernel

Radial Basis Function (RBF) or Gaussian Kernel

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$



σ^2 is known as the bandwidth

Can generalise to more general covariance matrices

Results in a Mercel kernel

Kernels on Discrete Data : Cosine Kernel

For text documents: let $\mathbf{x}$ denote bag of words

Cosine Similarity

$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x} \cdot \mathbf{x}'}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}$$

Kernels on Discrete Data : Cosine Kernel

For text documents: let $\mathbf{x}$ denote bag of words

Cosine Similarity

$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x} \cdot \mathbf{x}'}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}$$

Term frequency $\mathrm{tf}(c) = \log(1 + c)$, c word count for some word w

Inverse document frequency $\mathrm{idf}(w) = \log\left(\frac{N}{1+N_w}\right)$, N_w #docs containing w

$\text{tf-idf}(\mathbf{x})_w = \mathrm{tf}(x_w)\mathrm{idf}(w)$

Kernels on Discrete Data : String Kernel

Let $\mathbf{x}$ and $\mathbf{x}'$ be strings over some alphabet $\mathcal{A}$

$$\mathcal{A} = \{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$$

IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDYLQEFLGVMNTEWI

PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAERLQENLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKVLQELSQWTVRSIHDLRFISSHQTGIP

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_s w_s \phi_s(\mathbf{x}) \phi_s(\mathbf{x}')$$

$\phi_s(\mathbf{x})$ is the number of times s appears in $\mathbf{x}$ as substring

How to choose a good kernel?

Not always easy to tell whether a function is a Mercer kernel

For any finite set of inputs, the Kernel matrix should be positive definite

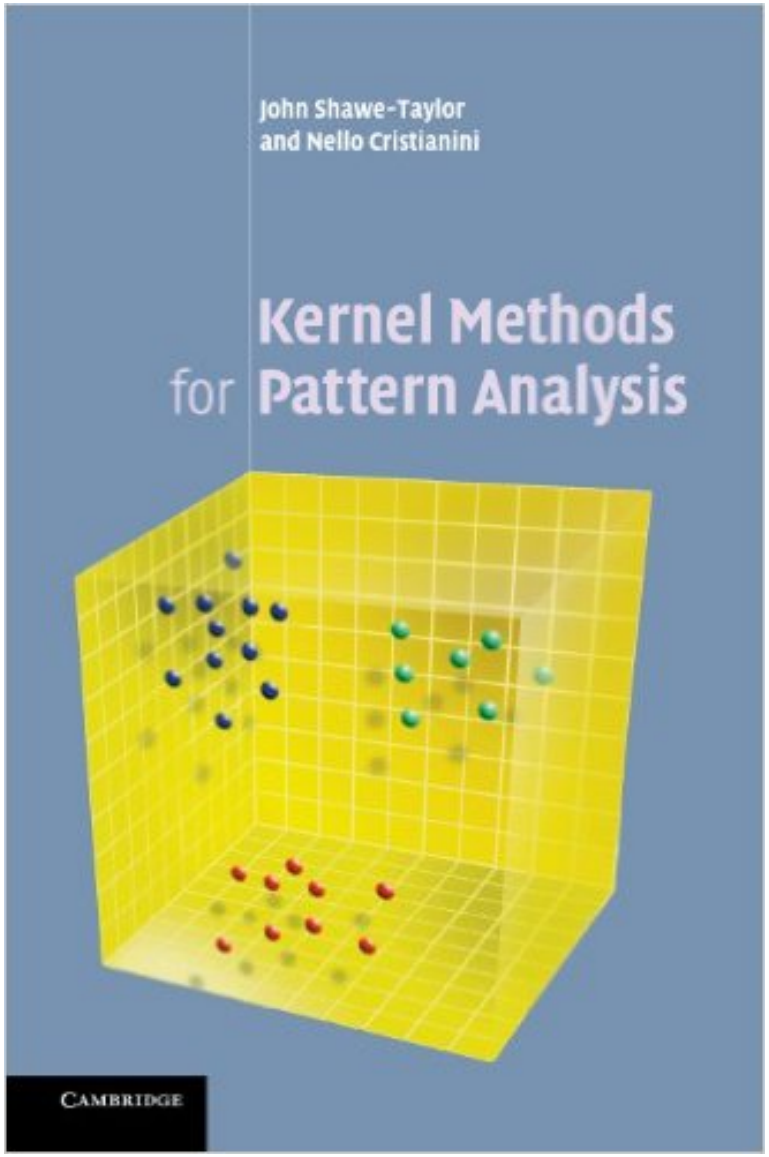
If the following hold:

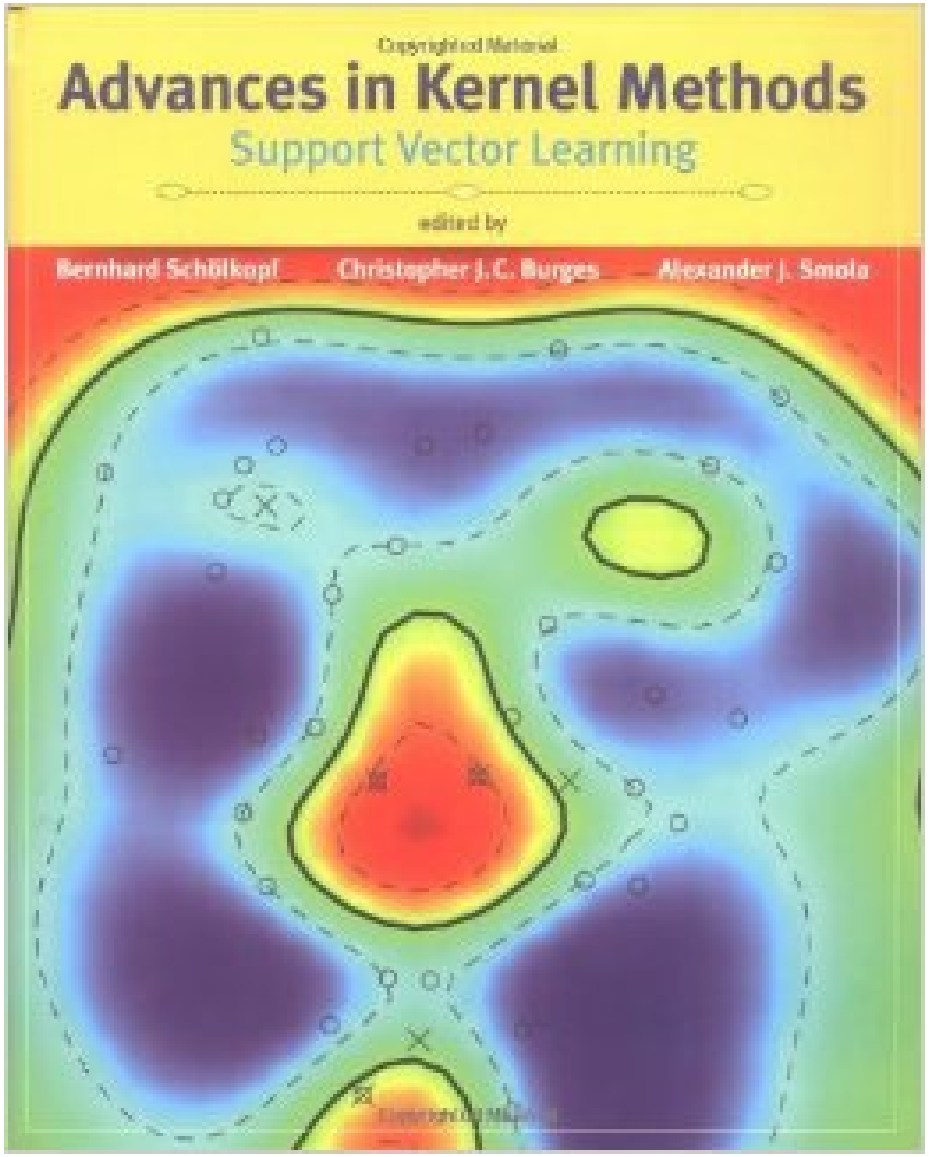
- κ_1, κ_2 are Mercer kernels for points in $\mathbb{R}^n$
- $f : \mathbb{R}^n \to \mathbb{R}$
- $\phi : \mathbb{R}^n \to \mathbb{R}^N$
- κ_3 is a Mercer kernel on $\mathbb{R}^N$

the following are Mercer kernels

- $\kappa_1 + \kappa_2$, $\kappa_1 \cdot \kappa_2$, $\alpha\kappa_1$ for $\alpha \in \mathbb{R}^+$
- $\kappa(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$
- $\kappa_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$
- $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{A}\mathbf{x}'$ for $\mathbf{A}$ positive definite

Books


John Shawe-Taylor
and Nello Cristianini
Kernel Methods
for Pattern Analysis
CAMBRIDGE


Advances in Kernel Methods
Support Vector Learning
edited by
Bernhard Schölkopf
Christopher J.C. Burges
Alexander J. Smola

Kernel Trick in Linear Regression

Recall the loss function for linear regression (least squares)

$$\mathrm{L}(\mathbf{w}) = \sum_{i=1}^{m} (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$

and the solution $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{y})$.

Can write $\hat{\mathbf{w}} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i$. Why?

Support Vector Regression

ϵ-sensitive loss function

$$\ell_\epsilon(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| < \epsilon \\ |y - \hat{y}| - \epsilon & \text{otherwise} \end{cases}$$

Loss function:

$$\mathrm{L}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \ell_\epsilon(\mathbf{x}_i^T \mathbf{w}, y_i)$$

Practical Next Week : SVM

MNIST digit classification using SVMs

Use `libsvm` in torch

Play using different kernels and tune C using cross validation

When using RBF kernels, need to tune $\gamma = \frac{1}{2\sigma^2}$ and C together

Grid Search:

$$\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$$
$$C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$$

Next Time : Neural Networks

- Online book: Michael Nielsen (www.michaelnielsen.org)
- Draft book: (www.deeplearningbook.org)