TU München
Fakultät für Informatik
PD Dr. Rudolph Triebel
John Chiotellis

Machine Learning for Robotics and Computer Vision
Winter term 2015

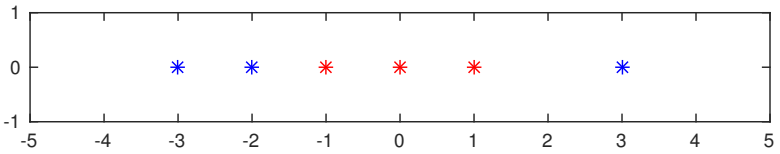
Solution Sheet 5

Topic: Boosting and Kernels
January 15th, 2016

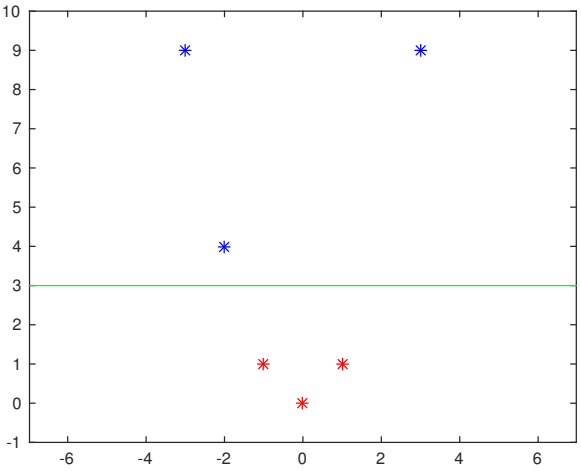
Exercise 1: Support Vector Machines

Consider a dataset with a single feature $x \in \mathbb{R}$ and labels $y \in \{+1, -1\}$. Data points $-3, -2, 3$ have label $+1$ and data points $-1, 0, 1$ have label -1.

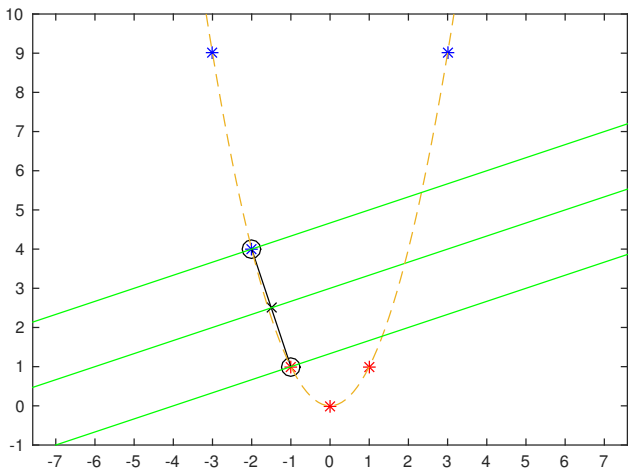
a) No, the dataset is not linearly separable. This becomes obvious once we plot the data points. There is no single line that can completely separate the two classes.



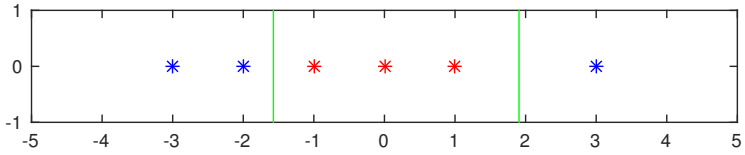
b) We can choose a feature map $\phi(x) = (x \quad x^2)^T \in \mathbb{R}^2$ so that the dataset becomes linearly separable. Now we can draw a line, actually infinitely many lines, that separate the two classes.



c) Construct a maximum-margin hyperplane and mark the support vectors.



The decision boundary in the original feature space corresponds to the x-coordinates of the intersections between the hyperplane and the feature function ($f(x) = x^2$). If we compute this exactly, the boundaries lie at $x = 1.9067$ and $x = -1.5734$. In the original space any point between these two would be classified as class -1, while all other points as class +1.



We can add any number of points to the training set without changing the hyperplane as long as the points lie "behind" the support vectors. This is because the separating hyperplane depends only on the support vectors. If we introduce a point that violates the margin, then this point should be considered a support vector and a new hyperplane should be computed.

Exercise 2: Gaussian Processes (Regression)

We want to approximate a function and we are given some samples (as training set) with input values $\mathbf{x}$ and outputs $\mathbf{y}$:

$$\mathbf{x} = (-0.8372, -0.4558, 0.6902, 0.1114, -0.4678)$$
$$\mathbf{y} = (-0.6414, -1.0286, -0.6893, -1.4021, -1.0594)$$

Suppose it has zero mean. Consider the kernel:

$$k(x_i, x_j) = \sigma_f^2 \exp(-\frac{1}{2l^2}(x_i - x_j)^2) + \sigma_n^2 \delta_{ij}$$

where $\sigma_f = 1$, $\sigma_n = 0.5$ and $\delta_{ij} = 1$, if $i = j$ and 0 otherwise.

a) Two new points $\mathbf{x}_* = (-0.5, 0.5)$ appear. Compute the mean value $\overline{\mathbf{y}_*}$ for $l = 1$.

We must first compute the Gram matrix (covariance of training points),

$$K = \begin{pmatrix} 1.25 & 0.92985 & 0.31146 & 0.63768 & 0.93405 \\ 0.92985 & 1.25 & 0.51858 & 0.85141 & 0.99993 \\ 0.31146 & 0.51858 & 1.25 & 0.84577 & 0.51146 \\ 0.63768 & 0.85141 & 0.84577 & 1.25 & 0.84558 \\ 0.93405 & 0.99993 & 0.51146 & 0.84558 & 1.25 \end{pmatrix}$$

Then we compute the Cholesky decomposition of K ($K = LL^T$),

$$L = \begin{pmatrix} 1.118 & 0 & 0 & 0 & 0 \\ 0.83168 & 0.7472 & 0 & 0 & 0 \\ 0.27858 & 0.38396 & 1.0124 & 0 & 0 \\ 0.57036 & 0.50463 & 0.48708 & 0.65787 & 0 \\ 0.83544 & 0.40834 & 0.12045 & 0.15862 & 0.58791 \end{pmatrix}$$

and the α vector as $\alpha = L^{-T}(L^{-1}y)$,

$$\alpha = \begin{pmatrix} 0.39321 \\ -0.16651 \\ 0.41008 \\ -1.2737 \\ -0.31433 \end{pmatrix}$$

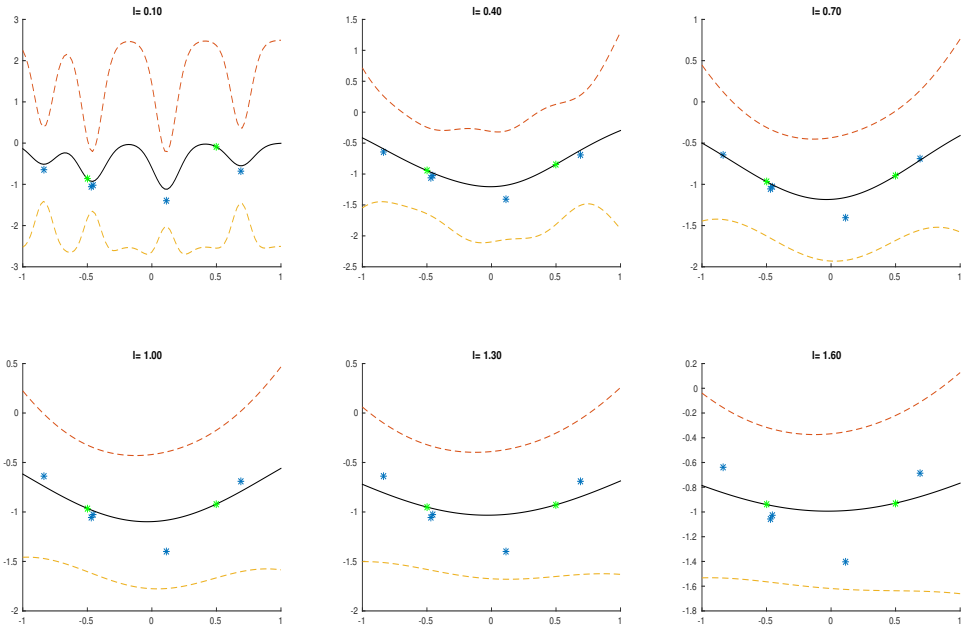
To evaluate the function on the test points, we first need to compute the matrix K_*, which is the kernel function applied between the test points and all training points,

$$K_* = \begin{pmatrix} 0.94473 & 0.409 \\ 0.99902 & 0.63332 \\ 0.49249 & 0.98207 \\ 0.82952 & 0.92728 \\ 0.99948 & 0.62605 \end{pmatrix}$$

Then our prediction is simply

$$\overline{\mathbf{y}_*} = K_*^T a = \begin{pmatrix} -0.9636 \\ -0.9198 \end{pmatrix}$$

b) Try different values for the hyperparameter l. Plot $\mathbf{y}_*$ against $\mathbf{x}_*$ with the confidence intervals (two standard deviations). How does the function change? Why?



As we can see, the function becomes smoother as we increase l. For small l the function fits better to the training data but cannot predict test data well. The variance is lower (confidence is higher) on the training points. As we increase l the function becomes smoother, it generalizes better and the variance becomes more uniform. If l gets too large, the function ignores the training points and does not generalize well anymore.

Exercise 3: Gaussian Processes (Programming)

See code.