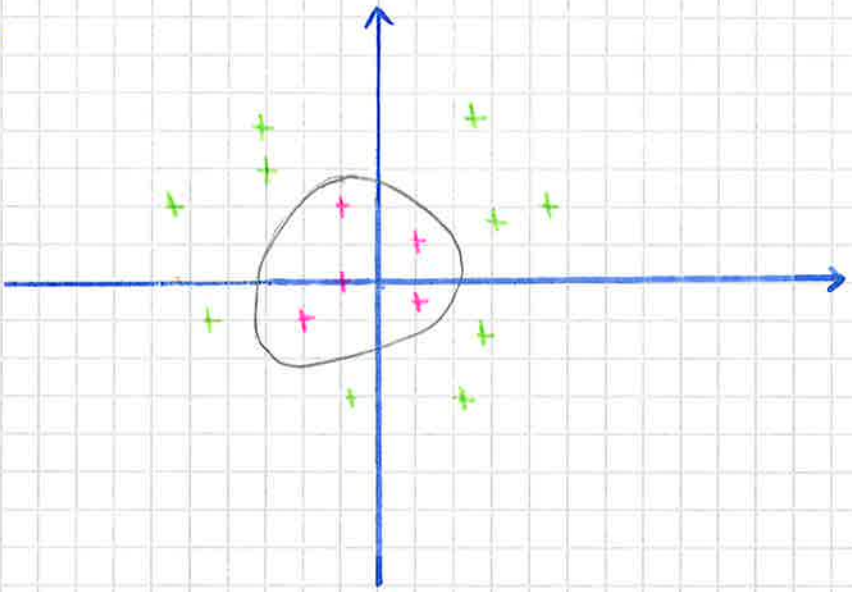
Chapter 5
Kernelized SVM's.

Example:



the data cannot be separated by a straight line. However, if, instead of representing each point by $x_n = [x_{n,1}; x_{n,2}]^T$, we represented them by their distance from the origin $\eta_n = x_{n,1}^2 + x_{n,2}^2$, we could separate them with a straight line. When we have a test point x_{new}, we compute η_{new} and classify it according to it being $\leq$ or $\geq$ to a certain value. In general, we use a transformation $\phi(x_n)$ of the n-th training object.

The most important characteristic of the SVM framework is that we never have actually to perform the transformation. In our objective and decision function, the data x_n, x_m, x_{new} appear within inner products ($x_n^T x_m$, $x_n^T x_{new}$, ...). After applying the transformation, we need to calculate the inner product in the new space ($\phi(x_n)^T \phi(x_m)$). So we do not need to think in term of transformation. Instead, if we can show that some function $k(x_n, x_m) = \phi(x_n)^T \phi(x_m)$ for some transformation ϕ, we are free to use $k(x_n, x_m)$ in place of any inner product. Function that corresponds to inner products in some space are known as kernel functions.

Our optimization and decision function (soft margin version) rewritten to include kernel functions are:

$$\begin{cases} \underset{\alpha}{\text{argmax}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_n \alpha_m t_n t_m k(x_n, x_m) \\ \text{subject to } \sum_{n=1}^{N} \alpha_n t_n = 0 \text{ and } 0 \le \alpha_n \le C, \forall n \end{cases}$$

$$t_{new} = \text{sign}\left(\sum_{n=1}^{N} \alpha_n t_n k(x_n, x_{new}) + b \right)$$

Most popular kernels:

- Linear: $k(x_n, x_m) = x_n^T x_m$
- Gaussian: $k(x_n, x_m) = \exp\{-\gamma (x_n - x_m)^T (x_n - x_m)\}$
- Polynomial: $k(x_n, x_m) = (1 + x_n^T x_m)^\gamma$

Linear is the one we used in Chapter 3. The Gaussian and polynomial kernels have a parameter that must be set

by the user (usually via cross-validation).

We draw a boundary obtained using a Gaussian kernel as the example. For the original SVM, we could compute the decision boundary exactly as it consisted of values of x that satisfied $w^T x + b = 0$. We can no longer compute w as it would be given by $\sum_n \alpha_n t_n \phi(x_n)$ and we do not necessarily know $\phi(x_n)$ (we only know $k(x_n, x_m) = \phi(x_n)^T \phi(x_m)$).
Therefore, to draw this decision boundary, we have had to evaluate $\sum_n \alpha_n t_n k(x_n, x_{new})$ over a grid of x_{new} values and then draw the contour corresponding to $\sum_n \alpha_n t_n k(x_n, x_{new}) = 0$.

Tuning γ

Modifying γ changes the (implicit) transformation $\phi(x_n)$, which will in turn change the kind of decision boundary. For the Gaussian kernel, increasing γ has the effect of increasing the complexity of the boundaries in the original space (mettre figures du livre de Rogers & Girolami, p.136).

↳ For γ "too big", the number of support vectors has increased dramatically so the solution can no longer considered sparse.

Kernelization

The SVM is not the only algorithm that can be kernelized.

Ex: KNN classifier

KNN requires the computation of the distances between x_{new} and each x_n. This distance can be

expressed as : $(x_{new} - x_n)^T (x_{new} - x_n)$

$$\| x_{new}^T x_{new} + x_n^T x_n - 2 x_{new}^T x_n .$$

We can replace this by a kernelized version:

$$k(x_{new}, x_{new}) - 2k(x_{new}, x_n) + k(x_n, x_n) .$$

This gives us a kernelized KNN.

Exercises: p. 32 - 104 of the python book