6.

Similarities:

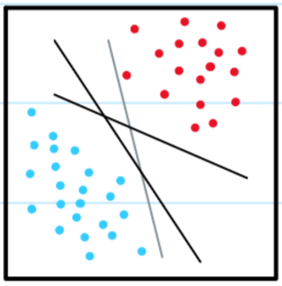
- Both algorithms are used to model linear and non-linear data sets.

- SVM does not support multiclass classification natively therefore we need to implement one of two indirect approaches: One-vs-one (1 model for each class pair) or One-vs-rest (where we consider one class and the remaining on a per model basis). Regarding the perceptron, we also require indirect modeling of this problem with the methods, in order to do so, being the same as in the SVM case.

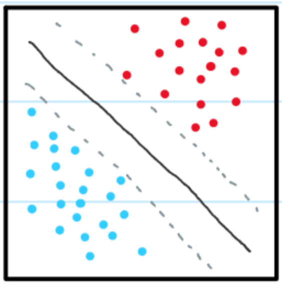
Differences

- SVM is based on a margin parameter, thus the algorithm stops after finding the best margin, whereas the perceptron algorithm stop when the data is classified correctly (converges on a set of weights).

- Main goal of the SVM is to separate data rigorously since a good margin avoids new samples falling on the wrong side of the boundary. The perceptron cannot compute this evaluation as precisely.



Perceptron, with multiple correct data divisions



Only possible SVM

- In non linear separable data, the Perceptron makes use of basis functions which extend linear regression models by allowing combinations of non-linear functions to better fit the data. Meanwhile, SVM introduces the kernel trick which simplifies working with basis functions. Some kernels are equivalent to using infinite-dimensional basis functions. While computing these feature transformations would be impossible, directly evaluating the kernels is often easy, making it more efficient. Kernels can also be used to encode similarity between arbitrary non-numerical data, from strings to graphs.

- Regarding optimization, the perceptron can be trained using back propagation, while SVM is solved by Sequential minimal optimization (SMO) for example, or any particular quadratic solver of the user's choice.

7.

$$g(\alpha) = \tfrac{1}{2}\,\alpha^T Q \alpha + \alpha^T \mathbb{1}_N$$

$$= \underbrace{-\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j}_{1} + \underbrace{\sum_{i=1}^{N}\alpha_i}_{2}$$

1 $$-\tfrac{1}{2}\underbrace{\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j}\cdot(x_i^T x_j)$$

$$= -\tfrac{1}{2}\cdot\sum_{i=1}^{N} y_i \alpha_i x_i\,(y_1\alpha_1 x_1 + y_2\alpha_2 x_2 + \ldots + y_N\alpha_N x_N)$$

$$= -\tfrac{1}{2}\cdot(y_1\alpha_1 x_1 + y_2\alpha_2 x_2 + \ldots + y_N\alpha_N x_N)^2 \in \mathbb{R}^{1\times 1}$$

Therefore, given that matrix product is commutative and the dual function SVM format:

$$Q = xx^T yy^T, \text{ with } y \in \mathbb{R}^{N\times 1},\ X \in \mathbb{R}^{N\times D}$$

b) As seen in the previous exercise (a), $\alpha^T Q \alpha$ results in a quadratic function (dependent on α). Given the concavity of such function, the local maximizer/minimizer is always a global maximum/minimum.

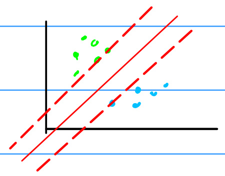
In order to determine the correct concavity (which must be negative) we need to determine the correct signal of Q, since $\alpha \geq 0$, for $i=1,\ldots,N$

Given $Q = \underbrace{(-1)}_{\leq 0}\cdot\underbrace{\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j x_i x_j}_{\geq 0}$, so $Q \leq 0$

$\downarrow$

negative semidefine matrix

③

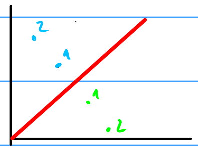


· an instance can only be missclassified if it forms a sv and it is removed from the training set, such that the next closest instance of the class holds

$y_k(w^T x + b) > m$ with m = margin size

(assuming the vector between the removed sv and the new one is perpendicular to the hyperplane on the full set, if this is not the case a similar argument can be made)

which implies that in the case where the next closest instance of each class holds $y_k(w^T x + b) > m$ we obtain $\epsilon = \frac{S}{N}$ (all sv are missclassified in LOOCV)



· by removing the sv 1 the new sv would be 2 but since the distance between 1 and 2 is greater than m the new hyperplane would be located above 1, missclassifying it as blue. with the symmetric argument for the sv 1, we can see that in this case all sv would be misclassified

if this is not the case and the next closest points are situated in a margin m around the sv then, no point would be missclassified as the shift in the hyperplane wouldn't be big enough. In this case $\epsilon < \frac{S}{N}$

$\Rightarrow \epsilon \leq S/N$

//

10 $k(x_1, x_2) = \sum_{i=1}^{N} a_i \left(x_1^T x_2\right)^i + a_0$ with $x_1, x_2 \in \mathbb{R}^d$

from the lecture:

- $x_1^T x_2 = x_1^T I x_2 \longrightarrow$ since I is a symmetric, PSD matrix $\Rightarrow k(x_1,x_2) = x_1^T x_2$ is a kernel
- $\sum_{i=1}^{N} k(x_1,x_2)$ is a kernel (adding kernels)
- $\sum_{i=1}^{N} k(x_1,x_2)^i$ is a kernel (multiplying kernels)
- $\sum_{i=1}^{N} a_i\, k(x_1,x_2)^i$ is a kernel (multiplying by non-negative constant)
- $\sum_{i=1}^{N} a_i\, k(x_1,x_2)^i + a_0$ is a kernel

11 $k(x_1, x_2) = \dfrac{1}{1 - x_1 x_2}$ with $x_1, x_2 \in (0,1)$

$k(x_1,x_2) := \phi(x_1)^T \phi(x_2)$

$\Rightarrow$ consider $\phi(x) = (1, x, x^2, \dots)$

$x_1 x_2 < 1$

$$k(x_1,x_2) = (1, x_1, x_1^2, \dots)\begin{pmatrix} 1 \\ x_2 \\ x_2^2 \\ \vdots \end{pmatrix} = 1 + x_1 x_2 + x_1^2 x_2^2 + \dots = \sum_{i=0}^{\infty} (x_1 x_2)^i = 1/(1 - x_1 x_2) //$$