OST
Ostschweizer
Fachhochschule

Lab 7

Support Vector Machines (SVM)

FTP MachLe MSE
HS 2023

MSE MachLe
WÜRC

Lernziele/Kompetenzen

- You know the primal form of a *separating hyperplane classifier*:

$$f_{w,b}(x) = \text{sign}(\langle w, x\rangle + b)$$

- You can calculate the *geometric margin* γ_i of a given sample point x_i for a separating hyperplane classifier $f_{w,b}$ with given parameters w and b.

$$\gamma_i := y_i \cdot \left(\frac{\langle w, x_i\rangle + b}{\|w\|}\right)$$

- You can explain the progression of the SVM family from *maximal margin classifier* to *support vector classifier* (SVC) and finally to *support vector machines* (SVM) that use the kernel trick.
- You can use SVM successfully on tutorial style examples, including (cross-validated) parameter *grid search*.
- You know, that SVM use only a subset of training points in the decision function (called *support vectors*), so they are memory efficient. SVM are still effective in cases where number of dimensions d is greater than the number of samples N.
- You know what a *kernel function* $K(x_i, x_j)$ is and how it is related to a *feature transform* $\phi(x)$ (Mercer Theorem). Different kernel functions can be specified for the decision function of a SVM. Common kernels are the *radial basis function* kernel, the *linear* kernel and the *polynomial* kernel.

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle_{\mathcal{H}}$$

- You can explain the effect of the C and γ parameters for a SVM with a radial basis function kernel (rbf-kernel). If you have a lot of noisy observations, C should be small. *Decreasing* C corresponds to *higher bias*, *less variance*, *larger margin*, *higher tolerance for misclassification*, less confidence in the dataset (more noise).

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

- You know that you can always *apply the kernel trick* whenever there appears an scalar product $\langle x_i, x_j\rangle$ of two feature vectors x_i and x_j in a given loss function $\mathcal{L}$. You know that by applying the kernel trick to the dual loss function of the separating hyperplane classifier linearly non-separable classes become separable if the chosen dimension is high enough.

1. Feature Transform ϕ of the Polynomial Kernel [A,I]

a) Wie calculate the feature transform $\phi(\vec{z}) : \mathbb{R}^2 \to \mathbb{R}$ associated with the inhomogeneous quadratic polynomial kernel $K(\vec{z}_1, \vec{z}_2)$ explicitely for two-dimensional features $\vec{z} = \begin{pmatrix} x \\ y \end{pmatrix}$, i.e. we have to find a feature transform $\phi(\vec{z})$ and an associated HILBERT space $\mathcal{H}$ such that the kernel can be expressed as a scalar product of the transformed features $\phi(\vec{z})$.

$$K(\vec{z}_1, \vec{z}_2) = (\vec{z}_1 \cdot \vec{z}_2 + 1)^2 = \langle \phi(\vec{z}_1), \phi(\vec{z}_2) \rangle_{\mathcal{H}}$$

We do this by expanding the square and then write down the kernel function as a *sum of symmetric products*. Then we are sure, that we can represent this as a scalar product in a higher dimensional vector HILBERT space $\mathcal{H}$. This is guaranteed by the MERCER theorem.

$$\begin{aligned}
K(\vec{z}_1, \vec{z}_2) &= (\vec{z}_1 \cdot \vec{z}_2 + 1)^2 \\
&= (\vec{z}_1 \cdot \vec{z}_2)^2 + 2\,(\vec{z}_1 \cdot \vec{z}_2) + 1 \\
&= (x_1 x_2 + y_1 y_2)^2 + 2 \cdot (x_1 x_2 + y_1 y_2) + 1 \\
&= (x_1^2 x_2^2 + 2 \cdot x_1 x_2 y_1 y_2 + y_1^2 y_2^2) + 2 \cdot (x_1 x_2 + y_1 y_2) + 1 \\
&= (x_1^2) \cdot (x_2^2) + (\sqrt{2} x_1 y_1) \cdot (\sqrt{2} x_2 y_2) + (y_1^2) \cdot (y_2^2) \\
&+ (\sqrt{2} x_1) \cdot (\sqrt{2} x_2) + (\sqrt{2} y_1) \cdot (\sqrt{2} y_2) + 1 \cdot 1
\end{aligned}$$

This is a sum of symmetric factors, where the first factor only depends on $\vec{z}_1$ and the second factor only depends $\vec{z}_2$. on By collecting now the first summands of the linearly independent compontens, we can construct the feature transform:

$$\phi(\vec{z}_1) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 y_1 \\ y_1^2 \\ \sqrt{2} x_1 \\ \sqrt{2} y_1 \\ 1 \end{pmatrix}$$

Whe can then write the kernel function $K(\vec{z}_1, \vec{z}_2)$ as a normal scalar product in a 6-dimensional vector HILBERT space.

$$K(\vec{z}_1, \vec{z}_2) = \langle \phi(\vec{z}_1), \phi(\vec{z}_2) \rangle_{\mathcal{H}} = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 y_1 \\ y_1^2 \\ \sqrt{2} x_1 \\ \sqrt{2} y_1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} x_2^2 \\ \sqrt{2} x_2 y_2 \\ y_2^2 \\ \sqrt{2} x_2 \\ \sqrt{2} y_2 \\ 1 \end{pmatrix}$$

b) The dimensionality $\dim(\mathcal{H})$ of the associated Hilbert space, i.e. the dimensionality of the transformed features $\phi(\vec{z})$ is $D = 6$.

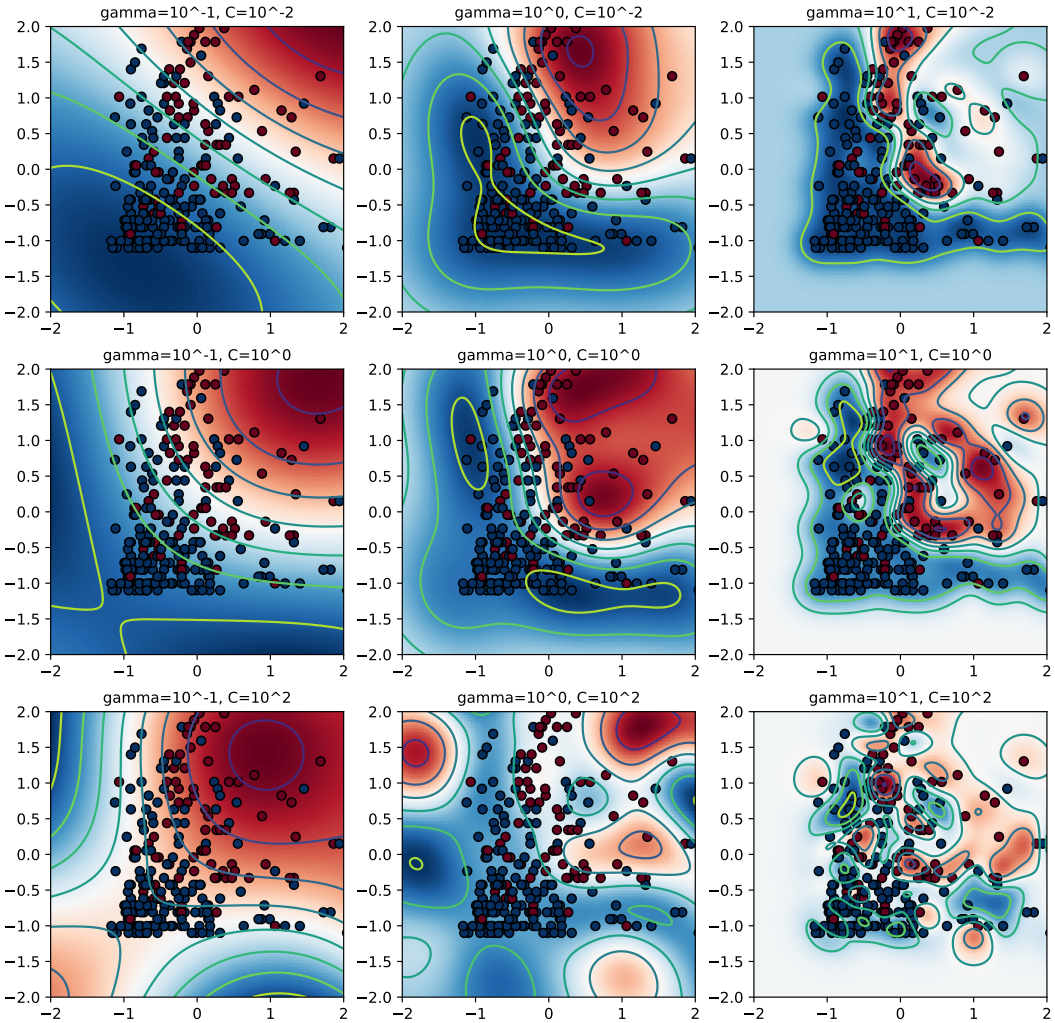
c) The dimensionality $\dim(\mathcal{H})$ of the associated Hilbert space for the radial basis function kernel would be $D = \infty$. We would have to expand the exponential function in an infinite Taylor series to be able to write down the kernel function as a normal dot product.

2. Simple SVM example [A,I]

The Juypter notebook can be found on moodle:
`ML07_A2_SVM_Simple.jpynb`

3. Predicting Diabetes using a SVM [A,II]

The Juypter notebook can be found on moodle:
`ML07_A3_PimaIndians.jpynb`



4. Polynomial Kernel SVM [A,I]
The Juypter notebook can be found on moodle:
`ML07_A4_PolynomialKernel.jpynb`

5. General Questions about SVM [A,I]

a) The fundamental idea behind Support Vector Machines is to fit the widest possible «street»between the classes. In other words, the goal is to have the *largest possible margin* between the decision boundary that separates the two classes and the training instances. When performing soft margin classification, the SVM searches for a compromise between perfectly separating the two classes and having the widest possible street (i.e., a few instances may end up on the street). Another key idea is to use *kernels* when training on nonlinear datasets.

b) After training an SVM, a support vector is any instance located on the «street»(see the previous answer), including its border. The decision boundary is entirely determined by the support vectors. Any instance that is not a support vector (i.e., off the street) has no influence whatsoever; you could remove them, add more instances, or move them around, and as long as they stay off the street they won't affect the decision boundary. Computing the predictions only involves the support vectors, not the whole training set.

c) SVMs try to fit the largest possible «street»between the classes (see the first answer), so if the training set is not scaled, the SVM will tend to neglect small features.

d) An SVM classifier can output the distance between the test instance and the decision boundary, and you can use this as a confidence score. However, this score cannot be directly converted into an estimation of the class probability. If you set `probability=True` when creating an SVM in `Scikit-Learn`, then after training it will calibrate the probabilities using Logistic Regression on the SVM's scores (trained by an additional five-fold cross-validation on the training data). This will add the `predict_proba()` and `predict_log_proba()` methods to the SVM.

e) This question applies only to linear SVMs *since kernelized can only use the dual form.* The computational complexity of the primal form of the SVM problem is proportional to the number of training instances N, while the computational complexity of the dual form is proportional to a number between N^2 and N^3. So if there are millions of instances, you should definitely use the primal form, because the dual form will be much too slow.

f) If an SVM classifier trained with an RBF kernel *underfits* the training set, there might be too much regularization. To decrease it, you need to *increase* γ or C (or both).