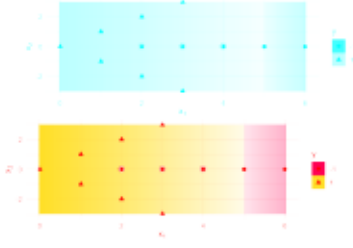
Non-linear Support Vector Machines

The Kernel Trick



Learning goals

- Know how to efficiently introduce non-linearity via the kernel trick
- Know common kernel functions (linear, polynomial, radial)
- Know how to compute predictions of the kernel SVM

Learning goals

- Know how to efficiently introduce non-linearity via the kernel trick
- Know common kernel functions (linear, polynomial, radial)
- Know how to compute predictions of the kernel SVM

DUAL SVM PROBLEM WITH FEATURE MAP

The dual (soft-margin) SVM is:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \left\langle \phi\left(\mathbf{x}^{(i)}\right), \phi\left(\mathbf{x}^{(j)}\right) \right\rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \\ & \sum_{i=1}^{n} \alpha_i y^{(i)} = 0, \end{aligned}$$

Here we replaced all features $\mathbf{x}^{(i)}$ with feature-generated, transformed versions $\phi(\mathbf{x}^{(i)})$.

We see: The optimization problem only depends on **pair-wise inner products** of the inputs.

This now allows a trick to enable efficient solving.

MERCER KERNEL

Definition: A **(Mercer) kernel** on a space $\mathcal{X}$ is a continuous function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

of two arguments with the properties

- Symmetry: $k(\mathbf{x}, \tilde{\mathbf{x}}) = k(\tilde{\mathbf{x}}, \mathbf{x})$ for all $\mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X}$.
- Positive definiteness: For each finite subset $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ the **kernel Gram matrix** $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ with entries $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ is positive semi-definite.

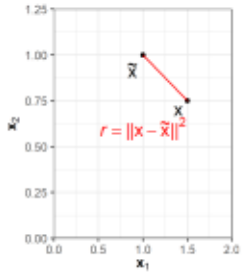
RBF KERNEL

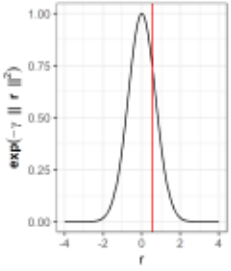
The “radial” **Gaussian kernel** is defined as

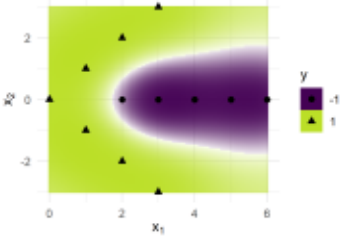
$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp(-\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{2\sigma^2})$$

or

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp(-\gamma \|\mathbf{x} - \tilde{\mathbf{x}}\|^2), \ \gamma > 0$$







KERNEL SVM

We kernelize the dual (soft-margin) SVM problem by replacing all inner products $\left\langle \phi\left(\mathbf{x}^{(i)}\right), \phi\left(\mathbf{x}^{(j)}\right)\right\rangle$ by kernels $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \left\langle \phi\left(\mathbf{x}^{(i)}\right), \phi\left(\mathbf{x}^{(j)}\right)\right\rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \\ & \sum_{i=1}^{n} \alpha_i y^{(i)} = 0. \end{aligned}$$

This problem is still convex because $\boldsymbol{K}$ is psd!

KERNEL SVM: PREDICTIONS

For the linear soft-margin SVM we had:

$$f(\mathbf{x}) = \hat{\theta}^T \mathbf{x} + \theta_0 \quad \text{and} \quad \hat{\theta} = \sum_{i=1}^{n} \alpha_i y^{(i)} \mathbf{x}^{(i)}$$

After the feature map this becomes:

$$f(\mathbf{x}) = \left\langle \hat{\theta}, \phi(\mathbf{x}) \right\rangle + \theta_0 \quad \text{and} \quad \hat{\theta} = \sum_{i=1}^{n} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)})$$

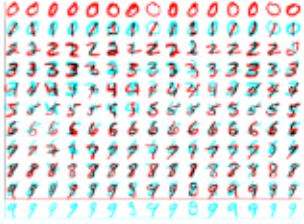
Assuming that the dot-product still follows its bi-linear rules in the mapped space and using the kernel trick again:

$$\left\langle \hat{\theta}, \phi(\mathbf{x}) \right\rangle = \left\langle \sum_{i=1}^{n} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}) \right\rangle = \sum_{i=1}^{n} \alpha_i y^{(i)} \left\langle \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}) \right\rangle =$$

$$= \sum_{i=1}^{n} \alpha_i y^{(i)} k(\mathbf{x}^{(i)}, \mathbf{x}), \qquad \text{so:} \qquad f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y^{(i)} k(\mathbf{x}^{(i)}, \mathbf{x}) + \theta_0$$

MNIST EXAMPLE

- Through this kernelization we can now conveniently perform feature generation even for higher-dimensional data. Actually, this is how we computed all previous examples, too.
- We again consider MNIST with 28×28 bitmaps of gray values.
- A polynomial kernel extracts $\binom{d+p}{d} - 1$ features and for the RBF kernel the dimensionality would be infinite.
- We train SVMs again on 700 observations of the MNIST data set and use the rest of the data for testing; and use C=1.



	Error
linear	0.134
poly (d = 2)	0.119
RBF (gamma = 0.001)	0.12
RBF (gamma = 1)	0.184