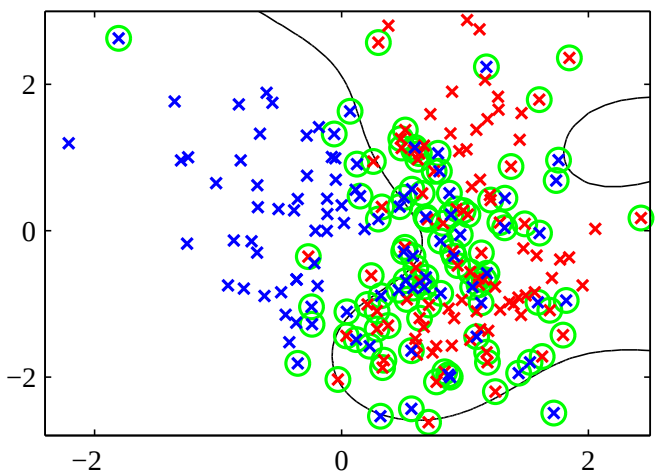
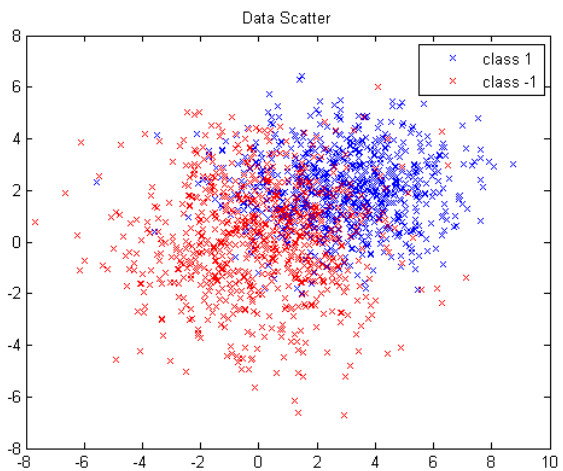
KERNELS

Classifiers discussed so far

- Both assume linear decision boundary
- Perceptron: Linear separability; placement of boundary rather arbitrary

More realistic data





Motivation: Kernels

Idea

- The SVM uses the scalar product $\langle \mathbf{x}, \tilde{\mathbf{x}}_i \rangle$ as a measure of similarity between $\mathbf{x}$ and $\tilde{\mathbf{x}}_i$, and of distance to the hyperplane.
- Since the scalar product is linear, the SVM is a linear method.
- By using a *nonlinear* function instead, we can make the classifier nonlinear.

More precisely

- Scalar product can be regarded as a two-argument function

$$\langle \,.\,,\,. \,\rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

- We will replace this function with a function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and substitute

$$k(\mathbf{x}, \mathbf{x}') \qquad \text{for every occurrence of} \qquad \langle \mathbf{x}, \mathbf{x}' \rangle$$

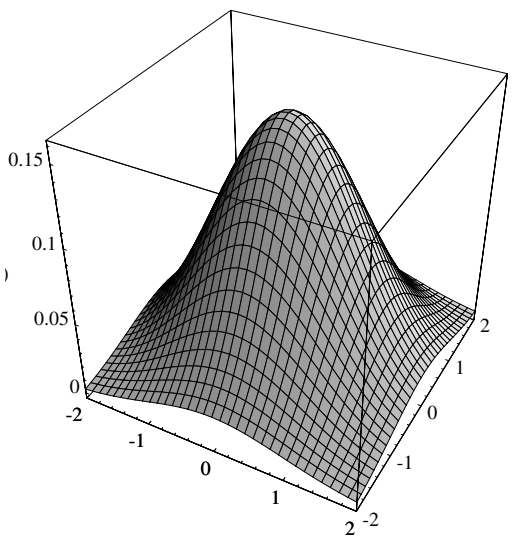
  in the SVM formulae.
- Under certain conditions on k, all optimization/classification results for the SVM still hold. Functions that satisfy these conditions are called **kernel functions**.

RBF Kernel

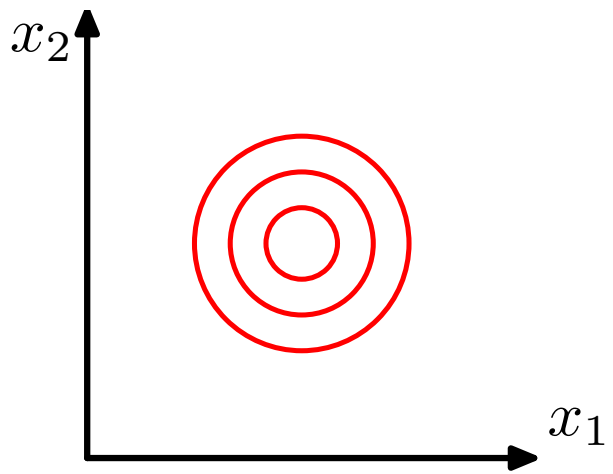
$$k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') := \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right) \qquad \text{for some } \sigma \in \mathbb{R}_+$$

is called an **RBF kernel** (RBF = radial basis function). The parameter σ is called **bandwidth**.
Other names for k_{RBF}: Gaussian kernel, squared-exponential kernel.

If we fix $\mathbf{x}'$, the function $k_{\mathrm{RBF}}(\,.\,, \mathbf{x}')$ is (up to scaling) a spherical Gaussian density on $\mathbb{R}^d$, with mean $\mathbf{x}'$ and standard deviation σ.



function surface (d=2)



contours (d=2)

Choosing a kernel

Theory

To define a kernel:

- We have to define a function of two arguments and prove that it is a kernel.
- This is done by checking a set of necessary and sufficient conditions known as "Mercer's theorem".

Practice

The data analyst does not define a kernel, but tries some well-known standard kernels until one seems to work. Most common choices:

- The RBF kernel.
- The "linear kernel" $k_{\text{SP}}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$, i.e. the standard, linear SVM.

Once kernel is chosen

- Classifier can be trained by solving the optimization problem using standard software.
- SVM software packages include implementations of most common kernels.

Which Functions work as Kernels?

Formal definition

A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a **kernel** on $\mathbb{R}^d$ if there is *some* function $\phi : \mathbb{R}^d \to \mathcal{F}$ into *some* space $\mathcal{F}$ with scalar product $\langle \,.\,,\,.\,\rangle_{\mathcal{F}}$ such that

$$k(\mathbf{x}, \mathbf{x}') = \big\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \big\rangle_{\mathcal{F}} \qquad \text{for all } \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d \ .$$

In other words

- k is a kernel if it can be interpreted as a scalar product on some other space.
- If we substitute $k(\mathbf{x}, \mathbf{x}')$ for $\langle \mathbf{x}, \mathbf{x}' \rangle$ in all SVM equations, we implicitly train a *linear* SVM on the space $\mathcal{F}$.
- The SVM still works: It still uses scalar products, just on another space.

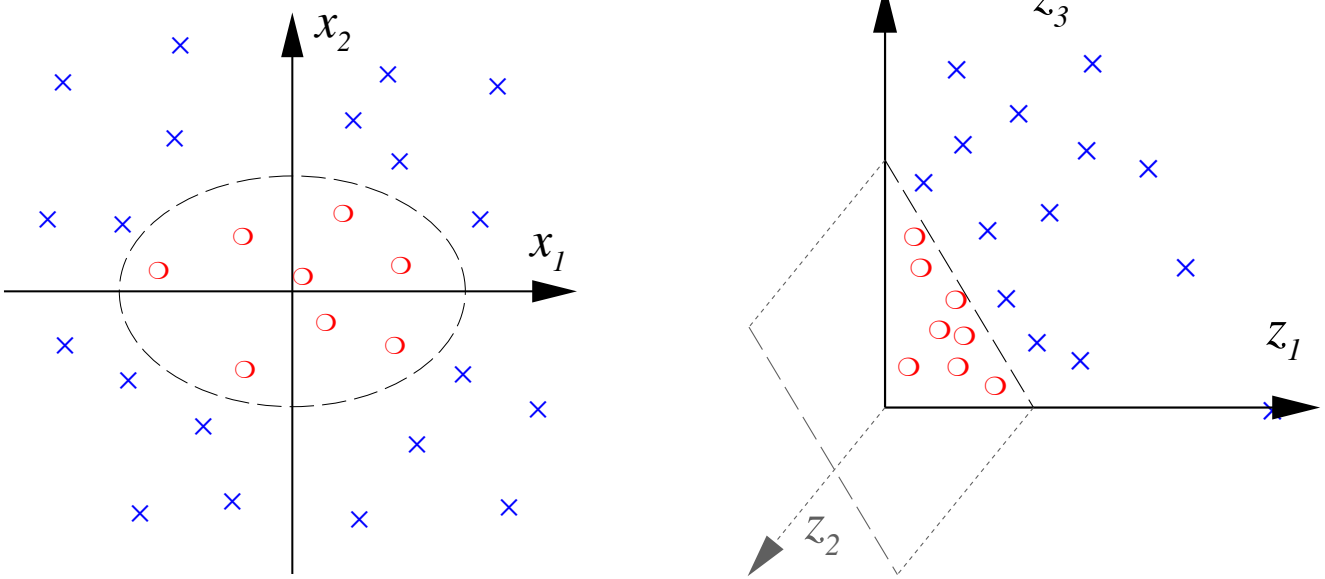
The mapping ϕ

- ϕ has to transform the data into data on which a linear SVM works well.
- This is usually achieved by choosing $\mathcal{F}$ as a higher-dimensional space than $\mathbb{R}^d$.

Example

How can a map into higher dimensions make class boundary (more) linear?
Consider

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3 \qquad \text{where} \qquad \phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \begin{pmatrix} x_1^2 \\ 2x_1x_2 \\ x_2^2 \end{pmatrix}$$



Mapping into Higher Dimensions

Problem

In previous example: We have to know what the data looks like to choose ϕ!

Solution

- Choose high dimension h for $\mathcal{F}$.
- Choose components ϕ_i of $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_h(\mathbf{x}))$ as different nonlinear mappings.
- If two points differ in $\mathbb{R}^d$, some of the nonlinear mappings will amplify differences.

The RBF kernel is an extreme case

- The function k_{RBF} can be shown to be a kernel, however:
- $\mathcal{F}$ is infinite-dimensional for this kernel.

DETERMINING WHETHER k IS A KERNEL

Mercer's theorem

A mathematical result called *Mercer's theorem* states that, if the function k is positive, i.e.

$$\int_{\mathbb{R}^d \times \mathbb{R}^d} k(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0$$

for all functions f, then it can be written as

$$k(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{x}') \ .$$

The ϕ_j are functions $\mathbb{R}^d \to \mathbb{R}$ and $\lambda_i \geq 0$. This means the (possibly infinite) vector $\phi(\mathbf{x}) = (\sqrt{\lambda_1}\phi_1(\mathbf{x}), \sqrt{\lambda_2}\phi_2(\mathbf{x}), \ldots)$ is a feature map.

Kernel arithmetic

Various functions of kernels are again kernels: If k_1 and k_2 are kernels, then e.g.

$$k_1 + k_2 \qquad\qquad k_1 \cdot k_2 \qquad\qquad \text{const.} \cdot k_1$$

are again kernels.

Kernels in general

- Many linear machine learning and statistics algorithms can be "kernelized".
- The only conditions are:
  1. The algorithm uses a scalar product.
  2. In all relevant equations, the data (and all other elements of $\mathbb{R}^d$) appear *only inside a scalar product*.
- This approach to making algorithms non-linear is known as the "kernel trick".

Optimization problem

$$\begin{aligned} \min_{\mathbf{v}_H, c} \quad & \|\mathbf{v}_H\|_{\mathcal{F}}^2 + \gamma \sum_{i=1}^{n} \xi^2 \\ \text{s.t.} \quad & y_i(\langle \mathbf{v}_H, \phi(\tilde{\mathbf{x}}_i) \rangle_{\mathcal{F}} - c) \geq 1 - \xi_i \quad \text{and } \xi_i \geq 0 \end{aligned}$$

Note: $\mathbf{v}_H$ now lives in $\mathcal{F}$, and $\| \, . \, \|_{\mathcal{F}}$ and $\langle \, . \, , \, . \, \rangle_{\mathcal{F}}$ are norm and scalar product on $\mathcal{F}$.

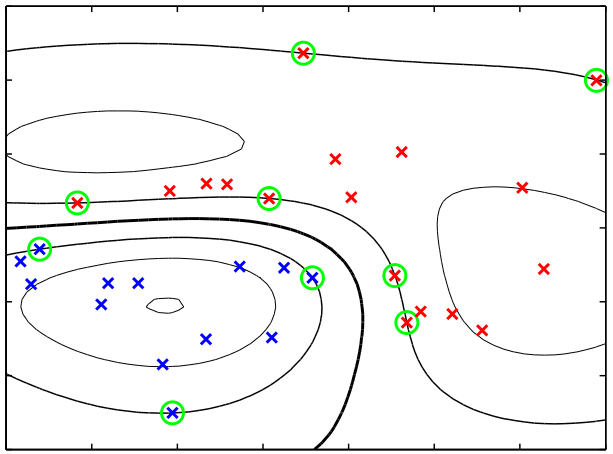
Dual optimization problem

$$\begin{aligned} \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad & W(\boldsymbol{\alpha}) := \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \tilde{y}_i \tilde{y}_j (k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) + \frac{1}{\gamma} \mathbb{I}\{i = j\}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0 \qquad \text{and} \qquad \alpha_i \geq 0 \end{aligned}$$

Classifier

$$f(\mathbf{x}) = \operatorname{sgn} \left(\sum_{i=1}^{n} \tilde{y}_i \alpha_i^* k(\tilde{\mathbf{x}}_i, \mathbf{x}) - c \right)$$

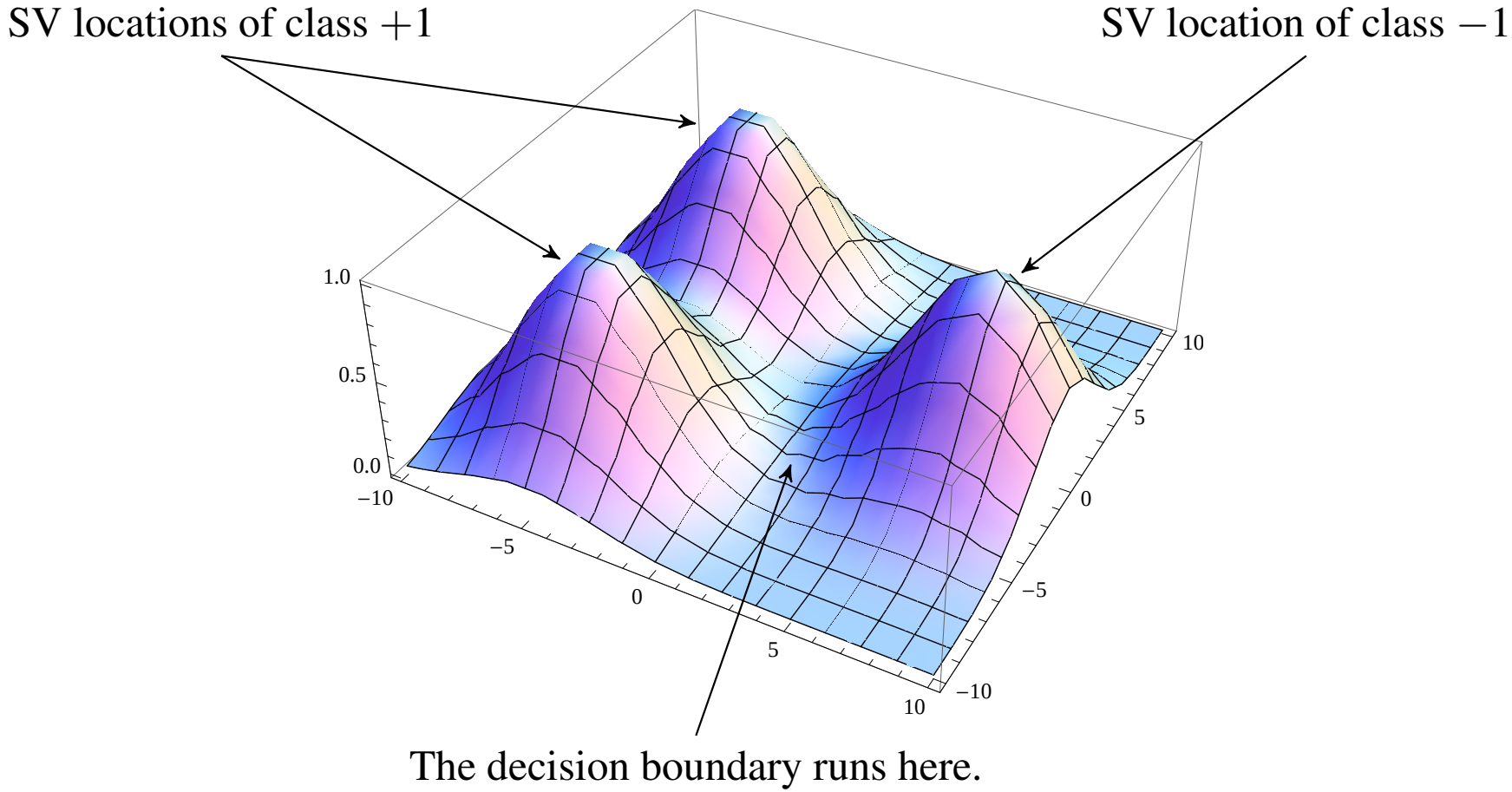
SVM WITH RBF KERNEL



$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} y_i \alpha_i^* k_{\mathrm{RBF}}(\mathbf{x}_i, \mathbf{x})\right)$$

- Circled points are support vectors. The the two contour lines running through support vectors are the nonlinear counterparts of the convex hulls.
- The thick black line is the classifier.
- Think of a Gaussian-shaped function $k_{\mathrm{RBF}}(\,.\,, \mathbf{x}')$ centered at each support vector $\mathbf{x}'$. These functions add up to a function surface over $\mathbb{R}^2$.
- The lines in the image are contour lines of this surface. The classifier runs along the bottom of the "valley" between the two classes.
- Smoothness of the contours is controlled by σ

Decision Boundary with RBF Kernel



The decision boundary of the classifier coincides with the set of points where the surfaces for class $+1$ and class -1 have equal value.

SUMMARY: SVMS

Basic SVM

- Linear classifier for linearly separable data.
- Positions of affine hyperplane is determined by maximizing margin.
- Maximizing the margin is a convex optimization problem.

Full-fledged SVM

Ingredient	Purpose
Maximum margin	Good generalization properties
Slack variables	Overlapping classes
	Robustness against outliers
Kernel	Nonlinear decision boundary

Use in practice

- Software packages (e.g. libsvm, SVMLite)
- Choose a kernel function (e.g. RBF)
- Cross-validate margin parameter γ and kernel parameters (e.g. bandwidth)

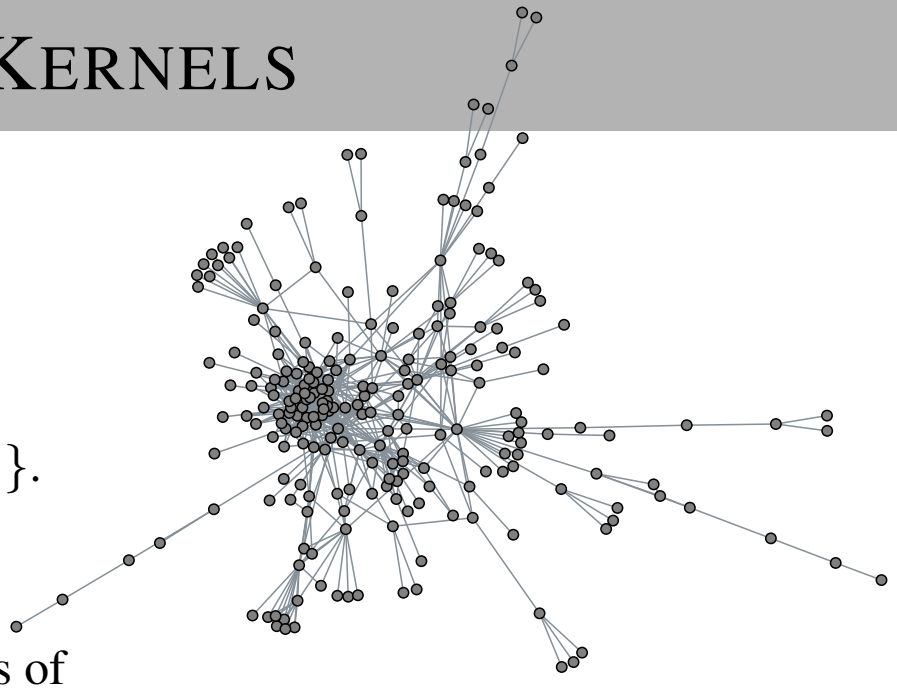
Unusual Example: Graph Kernels

Terminology

A **graph** $G = (V, E)$ is defined by two sets:

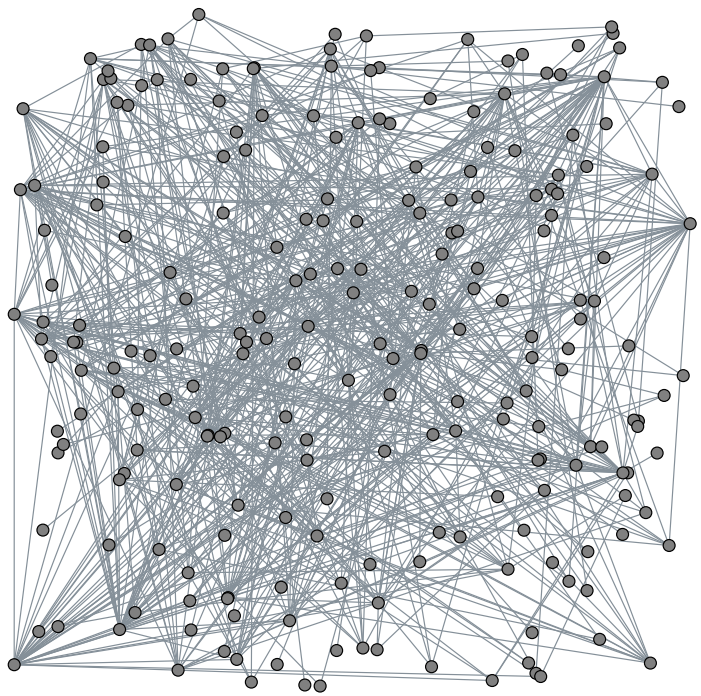
1. A set of V **vertices** $v_1, \ldots, v_m$.
2. A set E of **edges**, i.e. variables $e_{ij} \in \{0, 1\}$.

$e_{ij} = 1$ means that v_i and v_j are connected.



The graph is **undirected** if $e_{ij} = e_{ji}$ for all pairs of vertices. (The graphs in the figure are undirected.)

We write $\mathcal{G}$ for the set of undirected graphs of finite size.



Problem setting

- Training data $(\tilde{G}_i, \tilde{y}_i)_{i \in [n]}$, where each $\tilde{G}_i$ is a graph in $\mathcal{G}$.
- Can we learn a classifier f that classifies an unlabeled graph G?

Example 1: Social Networks

- Each vertex v_j is a user.
- $e_{ij} = 1$ indicates that users i and j are "friends".

This data is graph-valued, but the data set typically consists of a single, very large graph.

Example 2: Biology

There are dozens of types of graph-valued data in biology. One example is protein-protein interaction data:

- Each vertex v_j is a protein.
- $e_{ij} = 1$ indicates that proteins i and j interact in the given system.

(The graph on the previous slide shows such a data set.)

Graph kernels are designed for problems where we observe a set of graphs.

Modeling assumption

Graph G is characterized by how often certain patterns (= subgraphs) occur in G.

Feature map

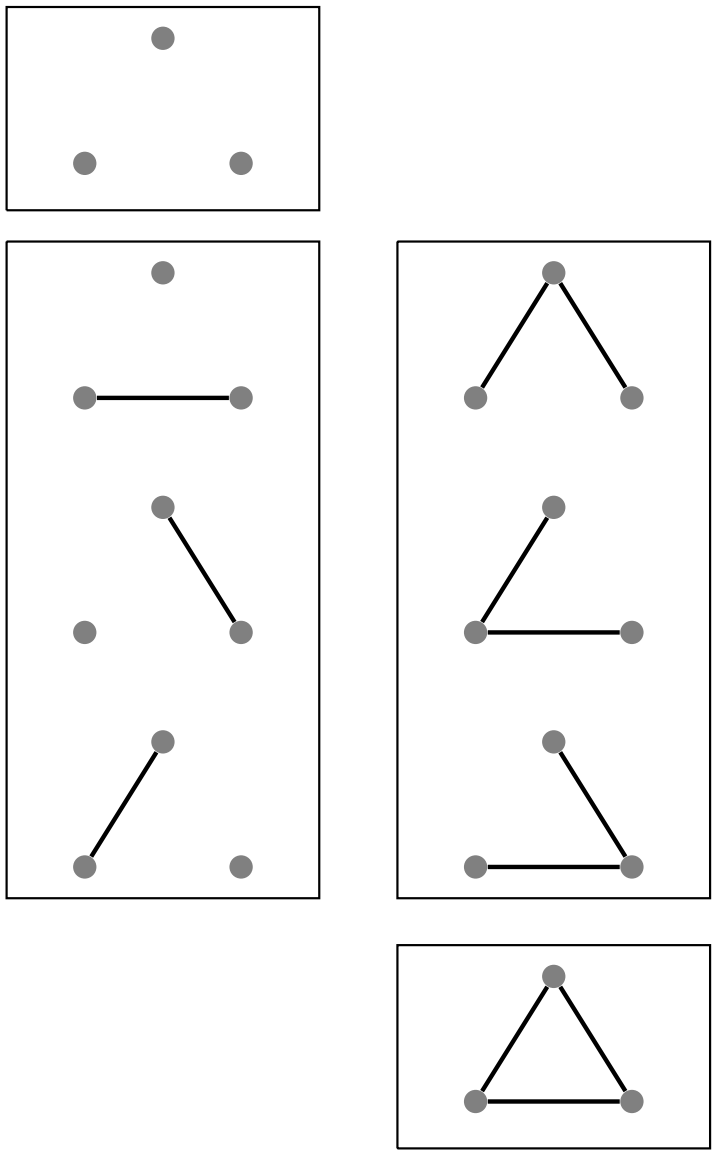
- Fix a set $\mathcal{K}$ of patterns.
  Example: All subgraphs of size 3.
- For graphs $G \in \mathcal{G}$, define

$$\phi_F(G) := \frac{\#\text{ occurrences of } F \text{ in } G}{\#\text{subgraphs of size } |F| \text{ in } G}$$

- Define the feature map ϕ as the vector

$$\phi(G) = (\phi_F(G))_{F \in \mathcal{K}}$$

  This is a mapping $\phi : \mathcal{G} \to \mathbb{R}_+^d$.
  The dimension is $d = |\mathcal{K}|$.



All subgraphs of size 3

Kernel

The kernel defined by ϕ is

$$k(G, G') := \left\langle \phi(G), \phi(G') \right\rangle = \sum_{F \in \mathcal{K}} \phi_F(G) \cdot \phi_F(G')$$

A large value of k indicates there is a subgraph in $\mathcal{K}$ that occurs often in *both* graphs.

Classification

We can now train an SVM as usual. For training data $(\tilde{G}_1, \tilde{y}_1), \ldots, (\tilde{G}_n, \tilde{y}_n)$, the resulting classifier is

$$f(G) = \operatorname{sgn}\left(\sum_{i=1}^{n} \tilde{y}_i \alpha_i^* k(\tilde{G}_i, G) - c \right)$$

Other graph kernels

- There are various other ways to define graph kernels. For example, k could compare G and G' in terms of the probability that a random walk on G and a random walk on G' take the same path. (Such kernels are called *random walk kernels*.)
- Each choice of kernel emphasizes a different property in terms of which the graphs are compared.

More generally: Kernels for non-Euclidean data

- We have used the kernel to transform non-Euclidean data (graphs) so that it fits into our classification framework.
- There are other, similar methods, e.g. string kernels.
- Note that we have not used the kernel for implicit representation, but rather compute ϕ explicitly.